# Privacy Preserving Clustering over Horizontal and Vertical Partitioned Data

Mina Sheikhalishahi, Fabio Martinelli
Istituto di Informatica e Telematica, Consiglio Nazionale delle Ricerche, Pisa, Italy
Email: name.surname@iit.cnr.it

***Abstract*—This paper presents a framework for constructing a hierarchical categorical clustering algorithm on horizontal and vertical partitioned dataset. It is assumed that data is distributed between two parties, such that for general benefits both are willing to detect the clusters on whole dataset, but for privacy concerns, they refuse to share the original datasets. To this end, we propose algorithms based on *secure weighted average protocol* and *secure number comparison protocol*, to securely compute the desired criteria in constructing clusters' scheme.**

***Index Terms*—Privacy, Hierarchical Clustering, Data Sharing, Secure Two-Party Computation, Distributed Clustering.**

## I. Introduction

Facing the new challenges brought by a continuous evolving Information Technologies (IT) market, large companies and small-to-medium enterprises found in *Information Sharing* a valid instrument to improve their key performance indexes. Sharing data with partners, authorities for data collection and even competitors, may help in inferring additional intelligence through collaborative information analysis [1] [2]. Such an intelligence could be exploited to improve revenues, e.g. through best practice sharing [3], market basket analysis [4], or prevent loss coming from brand-new potential cyber-threats [5]. Other applications include analysis of medical data, provided by several hospitals and health centers for statistical analysis on patient records, useful, for example, to shape the causes and symptoms related to a new pathology [6].

Independently from the final goal, unfortunately information sharing brings issues and drawbacks which must be addressed. These issues are mainly related to the information privacy. Shared information might be sensitive, potentially harming the privacy of physical persons, such as employee records for business applications, or patient records for medical ones. Hence, the most desirable strategy is the one which enables data sharing in secure environment, such that it preserves the individual privacy requirement while at the same time the data are still practically useful for analysis.

Clustering is a very well-known tool in unsupervised data analysis, which has been the focus of significant researches in different studies, spanning from information retrieval, text mining, data exploration, to medical diagnosis [7]. Clustering refers to the process of partitioning a set of data points into groups, in a way that the elements in the same group are more similar to each other rather than to the ones in other groups.

The problem of data clustering becomes challenging when data is distributed between two (or more) parties and for privacy concerns the data holders refuse to publish their own dataset, but still they are interested to shape more accurate clusters, identified on richer set of data.

To this end, we address the problem of securely constructing a hierarchical clustering algorithm, named CCTree, between two parties. CCTree (Categorical Clustering Tree) [8] has a decision tree-like structure, which iteratively divides the data of a node on the base of an attribute, or domain of values, yielding the greatest entropy. The division of data is represented with edges coming out from a parent node to its children, where the edges are labeled with the associated values. A node which respects the specified stop conditions, i.e. either pure enough or containing few elements, is considered as a leaf. The leaves of the tree are the desired clusters.

In this study, we consider two different scenarios of data being distributed either horizontally or vertically between two parties. In both scenarios, the participated parties are willing to model a CCTree on the whole dataset. For each scenario, we propose secure two-party computation protocols to verify whether in each new node of CCTree (including the root) the stop conditions are hold or not; and if not, which attribute respects the highest entropy for data division. At the end, each data holder finds the structure of CCTree on whole data, without knowing the records of other party.

In all this study it is assumed that clustering on joint datasets produces better result rather than clustering on individual dataset. In the following we present two motivating examples of present study:

• *Two-party vertical data distribution:* Suppose that two organizations, an *Internet marketing company* and an *on-line retail* company, own datasets with different attributes for a common set of records. These organizations are interested to share their data for clustering to find the optimal customer targets to maximize return on investments. How can these organizations learn about their clusters using each others data without learning anything about the attribute values of each other [9] ?

• *Two-party horizontal data distribution:* Suppose that a *hospital* and a *health center* hold different datasets with the same set of attributes. Both centers are interested to shape clusters on whole data, which brings the benefits of identifying the trends and patterns of diseases on the larger set of samples. How would it be possible to learn about clusters without disclosing patients' records [10]?

The contribution of this paper can be summarized as the following. A framework is proposed which serves as a tool for two parties to detect the cluster structures (based on CCTree) on the whole dataset, without revealing their data, in two different scenarios of data being partitioned either horizontally or vertically. Two secure computation protocols, named *secure weighted average* and *secure number comparison* protocols, are exploited to propose new algorithms such that each party is able to verify *stop conditions* and (if not satisfy) to find the *split attribute* for data division.

The rest of the paper is structured as follows. Related work on the two concepts of privacy preserving data clustering and secure decision tree construction is presented in Section II. Section III presents some preliminary notations exploited in this study, including *Categorical Clustering Tree (CCTree)*, *Secure Weighted Average Protocol*, and *secure Number Comparison Protocol*. Section IV describes the proposed framework, detailing the secure computation protocols for CCTree construction in both scenarios of data being either distributed horizontally or vertically. Finally Section V briefly concludes proposing future research directions.

## II. Related Work

The problem of privacy preserving data clustering is generally addressed for the specific case of $k$-means clustering, either when data is distributed between two parties [11], [12] or more than two parties [13]. Secure two-party $k$-means clustering is addressed in [11] based on *Paillier Homomorphic Encryption* scheme. Privacy preserving $k$-means clustering when data is distributed among more than two parties is addressed in [13], by proposing two protocols based on oblivious polynomial and homomorphic encryption, for computing cluster means. In [14], privacy preserving data clustering, when data is distributed horizontally among several parties is addressed through secure detection of *dissimilarity matrix*.

The more similar studies to what we proposed in present work can be found on secure decision tree construction, due to the fact that CCTree has decision tree like structure. In [15], the problem of secure ID3 classification is addressed, when data are vertically distributed. The paper does not address the problem when data are partitioned horizontally. Moreover, addressing a classification problem, it is required to access labeled data, differently from what we addressed in present work. As a parallel solution, in [16], the problem of constructing secure ID3 classifier is addressed when data are distributed horizontally. However, the problem is just discussed on horizontal partitioned data, and moreover, the data are labeled. In [17], the problem of secure ID3 classification is studied when data is distributed among several parties. The proposed solution is based on secure sum protocols. However, the proposed approach can not be applicable when two parties are involved.

In all aforementioned studies, differently from our approach, or data is distributed between more than two parties, or the problem is addressed for labeled data, or the number of clusters is known beforehand. To the best of out knowledge the problem of secure two-party data clustering is a topic which is required to be explored deeper.

## III. Preliminary Notations

In this section, we present some background knowledge which are exploited in our proposed framework.

### A. Categorical Clustering Tree (CCTree) Construction

CCTree is a hierarchical categorical clustering algorithm, constructed iteratively through a decision tree-like structure, where the leaves of the tree are the desired clusters. In contrast to decision tree, CCTree is constructed on unlabeled data. A simple example of CCTree is depicted in Figure 1.

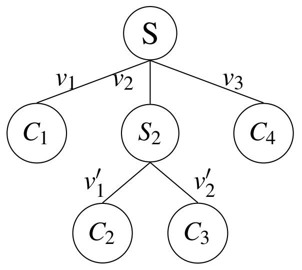

Fig. 1: A small CCTree

The root of the CCTree contains all the elements desired to be clustered. Each element is described through a set of *categorical* attributes. Being categorical, each attribute may assume a finite set of discrete values, constituting its domain. For example the attribute *Language* may have its domain as $\{English, French, Spanish\}$. At each new node of the tree (including the root), first two *stop conditions* are verified. If one of these stop conditions is hold, the node is labeled as *leaf*; Otherwise the best attribute is selected to generate new children nodes of the current node. The stop conditions for labeling a node as a leaf are as follows: 1) the number of elements in a node are less than a threshold, 2) the set of elements in a node are *homogeneous* enough. *Shannon Entropy* is the metric used both to define a homogeneity of a node, called *node purity*, and to select the best attribute to split a node. In particular non-leaf nodes are divided on the base of the attribute yielding the maximum value for Shannon entropy. The separation is represented through a branch for each possible outcome of the specific attribute. Each branch or edge extracted from parent node is labeled with the selected value which directs data to the child node. The process of CCTree construction can be formalized as follows.

**Input**: The input elements for constructing a CCTree are described in detail in the following:

*1) Attributes:* An ordered set of $k$ attributes $\mathcal{A} = \{A_1, A_2, \ldots, A_k\}$ is given, where each attribute is an ordered set of mutually exclusive values. Thus, the $i$'th attribute could be written as $A_i = \{v_1 \cdot A_i, v_2 \cdot A_i, \ldots, v_{|A_i|} \cdot A_i\}$, where $|A_i|$ is the number of values of attribute $A_i$, and $v_s \cdot A_i$ $(1 \leq s \leq |A_i|)$ represents $s$'th value of $i$'th attribute.

Let $n_j(v_s \cdot A_i)$ and $n(N_j)$ represents the number of elements in node $N_j$ that respect the $s$'th value of $i$'th attribute in node $N_j$. Considering $p(v_s \cdot A_i) = \frac{n_j(v_s \cdot A_i)}{n(N_j)}$, the *Entropy* of attribute $A_i$ in node $N_j$, denoted by $H(A_i, N_j)$, is defined as:

$$H(A_i, N_j) = -\sum_{s=1}^{|A_i|} p(v_s \cdot A_i) \log_2 p(v_s \cdot A_i) \tag{1}$$

*2) Data Points*: A set $D$ of $n$ data points is given, where each data point is a vector whose elements are the values of attributes, e.g. $x_i = (v_{i_1}, v_{i_2}, \ldots, v_{i_k})$, where $v_{i_j} \in A_j$. For example we may have a record as "ID1 = (Small, Red )".

*3) Stop Conditions:* A set of stop conditions $S = \{\mu, \varepsilon\}$ is given. $\mu$ is the "*minimum number of elements in a node*", i.e. when the number of elements in a node is less than $\mu$, then the node is not divided even if not pure enough. $\varepsilon$ represents the "*minimum desired purity*" for each cluster, i.e. when the purity of a node is less or equal to $\varepsilon$, it will be considered as a leaf. To calculate the node purity, a function based on Shannon entropy is defined as follows:
Let $n_j(v_s \cdot A_i)$ represents the number of elements having the $s$'th value of $i$'th attribute in node $N_j$, and $n(N_j)$ be the number of elements in node $N_j$. Considering $p_j(v_s \cdot A_i) = \frac{n_j(v_s \cdot A_i)}{n(N_j)}$, the *purity* of node $N_j$, denoted by $\rho(N_j)$, is defined as:

$$\rho(N_j) = -\frac{1}{k} \sum_{i=1}^{k} \sum_{s=1}^{|A_i|} p_j(v_s \cdot A_i) \log_2(p_j(v_s \cdot A_i)) \tag{2}$$

**Output**: The final output of the CCTree algorithm is a set of clusters, constituted by the leaves of the CCTree. For additional information on the CCTree algorithm we refer the reader to [18].

## *B. Secure Weighted Average Protocol (WAP)*

*Secure weighted average protocol (WAP)* is defined [19] to address the problem when *Alice* and *Bob* have $(a_1, a_2)$ and $(b_1, b_2)$, respectively. Both parties are interested to jointly compute $\frac{a_1+b_1}{a_2+b_2}$ in a secure way that each will know the final result without knowing the input of the other party.

To this end, let $(G, E, D, M)$ be an encryption scheme, where $G$ is the function for generating public parameters, $E$ and $D$ are the encryption and decryption functions, and $M$ is the message space, with the following properties:

- The encryption scheme $(G, E, D)$ is semantically secure, i.e. an adversary gains no extra information from inspecting ciphertext.
- For all $m, \alpha \in M$, $m_1 \in E(x)$ means that $m_1^{\alpha} \in E(m\alpha)$, i.e. $E(m)$ denotes the set of ciphertexts can be obtained by the encryption of $m$.
- There exists a computable function $f$ such that for all messages $m_1$ and $m_2$ we have $f(E(m_1), E(m_2)) = E(m_1 + m_2)$.

Keeping the above properties of our required probabilistic encryption system, the protocol of secure weighted average is presented as follows [19].

Assume that *Alice* sets up a probabilistic encryption scheme $(G, E, D, M)$ where the parameters of $G$ are public.

- (Step 1): *Alice* encrypts $a_1$ and $a_2$ and sends the encrypted values $a_1^{'} \in E(a_1)$ and $a_2^{'} \in E(a_2)$ to *Bob*.
- (Step 2): *Bob* computes a random message $m \in M$ and encrypts $m \cdot b_1$ and $m \cdot b_2$ to obtain $m_1' \in E(m \cdot b_1)$ and $m_2' \in E(m \cdot b_2)$. Then, *Bob* calculates $\mathcal{M}_1 = f(a_1'^{m}, m_1')$, $\mathcal{M}_2 = f(a_2'^{m}, m_2')$, and sends the result to *Alice*, where $f$ is computed through multiplication).
- (Step 3): From the properties of probabilistic scheme $(G, E, D)$, the following are obtained:
  $\mathcal{M}_1 = E(m \cdot a_1 + m \cdot b_1)$ , $\mathcal{M}_2 = E(m \cdot a_2 + m \cdot b_2)$
  Hence, *Alice* is able to compute $m \cdot (a_1 + b_1)$ and $m \cdot (a_2 + b_2)$ and consequently $\frac{a_1+b_1}{a_2+b_2}$. *Alice* sends the final result to *Bob*.

In the rest of this study we denote the call of this protocol by *WAP*($(a_1, a_2), (b_1, b_2)$ ), which means that two pairs of $(a_1, a_2)$ and $(b_1, b_2)$ are hold by two different parties, and they both get the weighted average result, without knowing each other data.

## *C. Secure Numbers Comparison Protocol*

*Alice* and *Bob* have natural numbers $N_a$ and $N_b$, respectively. They want to know whose number is greater than the other one's without revealing the numbers. To address this problem, we present and exploit the protocol proposed in [20]. Let $1 < N_a, N_b < N$, $\mathcal{M}$ be the set of all $N$-bit nonnegative integer numbers, and $Q_N$ be the set of all one-to-one functions from $\mathcal{M}$ to $\mathcal{M}$. Moreover, suppose $E_a$ be the public key of *Alice* generated by a random variable from $Q_N$. Then, the protocol proceeds as the following:

- (Step 1): *Bob* selects a random $N$-bit integer, and computes the encrypted value $x' = E_a(x)$
- (Step 2): *Bob* sends *Alice* the number $x' - N_b + 1$
- (Step 3): *Alice* computes $x^d = D_a(x' - N_b + u)$ for $u = 1, \ldots, N$.
- (Step 4): *Alice* generates a random prime number $p$ of $\frac{N}{2}$-bits and calculates $Z_u = x^d (\mod p)$ for all $u$; if all $z_u$ are different from each other by at least 2 in the $\mod p$ sense, stop; otherwise generates another random prime number and repeat the process until all $z_u$ differ by at least 2. Without loss of generality, let $p$ and $z_u$ be the final results.
- (Step 5): *Alice* sends the prime number $p$ and the following $N$ numbers $z_1, \ldots, z_N$ to *Bob*.
- (Step 6): *Bob* looks the $N_b$'th (not containing $p$) sent from *Alice*, and decides that $N_a \geq N_b$ if it is equal to $x \mod p$, and $N_a < N_b$ otherwise.
- (Step 7): *Bob* tell to *Alice* the conclusion.

In the rest of this study, we denote the call of *secure number comparison* protocol as a boolean function *SNC* which gets two natural numbers $N_a$ and $N_b$, provided by *Alice* and *Bob* respectively; it returns 1 if $N_a \geq N_b$, and 0 otherwise. Formally,

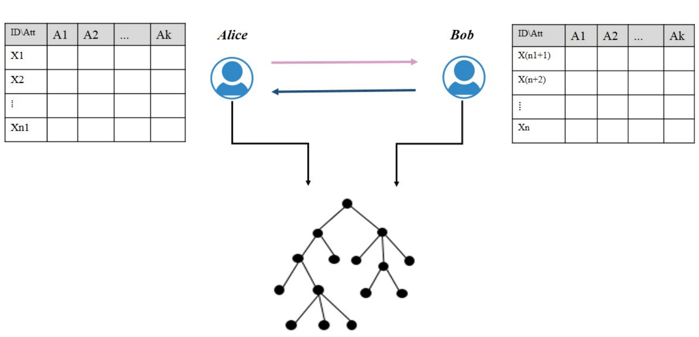

Fig. 2: CCTree Construction on Horizontal Distributed Data

we have:

$$SNC(N_a, N_b) = \begin{cases} 1 & N_a \geq N_b \\ 0 & N_a < N_b \end{cases}$$

## IV. Framework Modeling

Suppose two data holders are interested to detect the structure of clusters (through CCTree) on the whole of their datasets. However, for privacy concerns, they are not willing to publish or share the main dataset. As mentioned before, it is assumed that clustering on the whole of both datasets (as in general cases) produces the better result comparing to clustering on individual dataset.

In what follows, we consider two scenarios of the problem, i.e. when data is divided horizontally and vertically between two parties. To this end, in both scenarios, we propose the protocols which verifies the stop conditions in each new node of CCTree, and if stop conditions are not hold, then the optimum attribute is found for data division. It is assumed that the two participated parties set together the stop conditions, i.e. node purity ($\varepsilon$) and minimum number of elements ($\mu$), before beginning the communication. Thence, the structure of CCTree is shaped on whole data, without revealing data to other party.

In Sections IV-A and IV-B, we detail the process of CCTree construction on horizontal and vertical partitioned data, respectively. It is assumed that both participated parties are *honest*, and they completely follow the protocols.

### *A. Horizontal Data Distribution*

Let consider that two data holders are interested to model a CCTree on the whole dataset when data are distributed horizontally. This means that each data holder has information about all the features but for different collection of objects. More precisely, let $\mathcal{A} = \{A_1, A_2, \ldots, A_k\}$ be the set of $k$ categorical attributes all used to express each record of data. Therefore, each record is a $k$ dimensional vector $x_i = (v_{i_1}, v_{i_2}, \ldots, v_{i_k})$, where $v_{i_j} \in A_j$.

Figure 2 depicts a higher level representation of CCTree construction on horizontal distributed framework. *Alice* and *Bob*, holding respectively datasets $D_a$ and $D_b$, are the two parties interested in constructing CCTree on $D_a \cup D_b$, without knowing the data information of the other party. As it can be observed, the two tables are described with the same set of attributes, but on different objects. To build the CCTree on whole data, *Alice* and *Bob* communicate trough upcoming secure algorithms, in each new node of the CCTree, to detect if the node satisfies the stop conditions (to be labeled as a leaf), and if not, which attribute respects the highest entropy for data division. At the end, both parties get the final CCTree structure identified on whole data, without revealing data to the other party.

*1) Attribute Entropy Computation:* In each new node of the CCTree (including the root), it is required to securely compute the Shanon entropy of each attribute, for both verifying the node purity, and for data division (if it is not a leaf). In what follows, we first discuss in detail how securely *Alice* and *Bob* are able to compute *attribute entropy*, when the attribute constitutes of *merely* two values. Afterwards, in Algorithm 1 we generalize the process of secure attribute entropy computation for any length of attribute.

Suppose for a specific attribute $A$, *Alice* and *Bob* hold the information $A^a = (a_1, a_2)$ and $A^b = (b_1, b_2)$, respectively. For instance attribute $A$ can be considered as the *Size* (= $(small, large)$) of records, such that *size* statistical information of *Alice* dataset is recorded by $A^a = (a_1, a_2)$ = (*number of Alice records in small size*, *number of Alice records in large size* ). The same information about the *size* of data is computed on *Bob* dataset and saved as $A^b = (b_1, b_2)$.

*Alice* and *Bob* are required to compute *securely* the *entropy* of attribute $A$, denoted by $H(A)$, as the following:

$$H(A) = -\left(\left(\frac{a_1+b_1}{a_1+a_2+b_1+b_2}\right)\cdot \log_2\left(\frac{a_1+b_1}{a_1+a_2+b_1+b_2}\right) + \left(\frac{a_2+b_2}{a_1+a_2+b_1+b_2}\right)\cdot \log_2\left(\frac{a_2+b_2}{a_1+a_2+b_1+b_2}\right)\right)$$

Referring to the weighted average protocol presented in Section III-B, we know when *Alice* has $(a_1, a_2)$ , and *Bob* has $(b_1, b_2)$, how *securely* the weighted average $\frac{a_1+b_1}{a_2+b_2}$ is obtained. In the following steps we show how simply from the application of secure weighted average protocol both *Alice* and *Bob* securely compute $H(A)$:

(Step 1): *Alice* and *Bob* compute in their own side $\alpha = a_1 + a_2$ and $\beta = b_1 + b_2$, respectively, without revealing the result.

(Step 2): From the secure protocol presented in III-B, *Alice* and *Bob* can obtain securely the weighted average of $(a_1, \alpha)$ and $(b_1, \beta)$, and also the weighted average of $(a_2, \alpha)$ and $(b_2, \beta)$ as well. This means that both *Alice* and *Bob* securely compute:

$$\mathcal{W}_1 = \frac{a_1+b_1}{\alpha+\beta} = \frac{a_1+b_1}{a_1+a_2+b_1+b_2}\ ,\ \mathcal{W}_2 = \frac{a_2+b_2}{\alpha+\beta} = \frac{a_2+b_2}{a_1+a_2+b_1+b_2}$$

(Step 3): Having the results of Step 2, both *Alice* and *Bob* compute in their own side:

$$H(A) = -\mathcal{W}_1 \cdot \log_2(\mathcal{W}_1) - \mathcal{W}_2 \cdot \log_2(\mathcal{W}_2)$$

Algorithm 1 generalizes the process of *attribute entropy computation* in the case that the length of attribute $A$ might be in any length, say $t$. Due to the fact that attribute entropy is computed in each new node of CCTree, the node label is also specified in the algorithm.

**Algorithm 1:** *Att.Entropy()*: Secure Attribute Entropy Computation

```
Data: Alice and Bob have statistical information of
      attribute $A$ as $A^a = (a_1,\dots,a_t)$, $A^b = (b_1,\dots,b_t)$ in
      node $N_j$, respectively.
Result: Alice and Bob compute securely
      $H(A,N_j) = -\sum_{i=1}^{t}(\frac{a_i+b_i}{a_1+\dots+a_t+b_1+\dots+b_t} \cdot \log_2(\frac{a_i+b_i}{a_1+\dots+a_t+b_1+\dots+b_t}))$
1 initialization;
2 $\mathcal{W}_i, \mathcal{W} = 0$
3 Alice:  $\alpha \leftarrow \sum_{i=1}^{t} a_i$
4 Bob:    $\beta \leftarrow \sum_{i=1}^{t} b_i$
5 for $1 \le i \le t$ do
6     $\mathcal{W}_i \leftarrow$ WAP ( $(a_i,\alpha), (b_i,\beta)$ )
7     $\mathcal{W} = \mathcal{W} - (\mathcal{W}_i \cdot \log_2(\mathcal{W}_i))$
8 end
9 return $\mathcal{W}$
```

**Theorem IV.1.** *Algorithm 1 reveals nothing to any party except the entropy of an attribute.*

*Proof.* The only communications between *Alice* and *Bob* occur at line 6 which consist of a call to the *secure weighted average protocol*, proven to be secure in [19]. Since in the rest of algorithm, there is no communication between two participated parties, the algorithm will not violate parties' privacy. □

Applying the above algorithm, *Alice* and *Bob* obtain the entropies of all attributes without knowing the extra information of the other party's dataset. The attribute with the highest entropy is the optimum one for data division (if node is not already labeled as a leaf). In what follows, we propose secure two-party computation algorithms for checking the stop conditions in each new node of CCTree.

*2) Stop condition (node purity):* In a node of CCTree, when the data constituting the node are *pure enough*, the node is labeled as a leaf (cluster). Before CCTree construction, *Alice* and *Bob* set and fix together the required threshold of node purity, say $\varepsilon$. The formula for *node purity* computation has been defined in Equation 2. Algorithm 2 details the process.

**Algorithm 2:** *Node.purity()*: Secure Node Purity Computation

```
Data: Alice and Bob have statistical information of node
      $N_j$, as $N_j^a$ and $N_j^b$ respectively.
Result: Alice and Bob compute securely $\rho(N_j)$
1 initialization;
2 $\rho(N_j) = 0$
3 for $1 \le i \le k$ do
4     Att.Entropy($A_i^a, A_i^b, N_j$)
5     $\rho(N_j) = \frac{1}{k}(\rho(N_j)$ + Att.Entropy($A_i^a, A_i^b, N_j$) )
6 end
7 return $\rho(N_j)$
```

**Theorem IV.2.** *Algorithm 2 reveals only the entropy of attributes and nothing about the counts of values in each party's dataset.*

*Proof.* The only communication between *Alice* and *Bob* occurs at line 4, which is a call to the algorithm 1, proven to be secure in Theorem IV.1. □

If *Alice* and *Bob* find that the purity of a specific node is less than the purity threshold $\varepsilon$, then that node is labeled as a leaf; Otherwise the other stop condition, i.e. number of elements in a node is verified through Algorithm 3. If both are not hold, the best attribute for data division is found through Algorithm 1. i.e. the attribute with the highest entropy.

*3) Stop condition (number of elements):* In CCTree construction, one of the stop conditions for labeling a node as a leaf, is that the number of elements in the node be less than a specified threshold. To this end, before constructing the CCTree, *Alice* and *Bob* set together the minimum number of elements in a node to be identified as a *leaf*, say $\mu$. Algorithm 3 gives the details of secure verification of this stop condition.

**Algorithm 3:** *Num.elements()*: Secure Number of Elements Validation

```
Data: Alice and Bob have private numbers $N_a$ and $N_b$,
      respectively; $\mu$ is a public number.
Result: Alice and Bob know securely whether
      $N_a + N_b \le \mu$ or not.
1 initialization;
2 Bob computes $\mu - N_b$
3 if $\mu - N_b \le 0$ then
4     return 1
5 else
6     return SNC ($N_a$ , $\mu - N_b$ )
7 end
```

If the answer is 0, then the current node is labeled as a "*leaf*"; Otherwise, if also the node is pure enough through the result of Algorithm 2, the best attribute for data division is found from the comparisons of attribute entropies computed securely through Algorithm 1.

**Theorem IV.3.** *Algorithm 3 only finds if the number of elements in a node is less than a threshold, and the information of records are not revealed.*

*Proof.* Since the proposed algorithm is based on secure number comparison protocol and the underlying protocol has been proven to be secure in [20], the proposed approach reveals nothing except if the addition of two numbers is less than a threshold. □

With the use of Algorithms 1, 2, and 3, *Alice* and *Bob* are able to get a unique CCTree on the whole of their datasets, without revealing their original datasets. Algorithm 4 captures the overall process.

**Algorithm 4:** *CCTree.Horizontal()*: Secure Two-Party CCTree Construction, Horizontal Data Distribution

```
Data: Alice holds $D_a$, Bob holds $D_b$, on set of attributes
      $\{A_1, A_2, \dots, A_k\}$, stop conditions $\mu$ and $\varepsilon$
Result: Secure (horizontal) CCTree construction on
      $D = D_a \cup D_b$
1  initialization;
2  Root node $N_0$ contains all $D$
3  for node $N_j \neq leaf$ do
4      if (Node.purity($N_j^a, N_j^b$) $< \varepsilon$
5      or Num.elements ($n(N_j^a), n(N_j^b), \mu$)) = 0 then
6          Label $N_j$ as leaf
7      else
8          for $1 \leq i \leq k$ do
9              $Att.Entropy(A_i^a, A_i^b, N_j)$
10             $A^* \leftarrow argmax_i Att.Entropy(A_i^a, A_i^b, N_j)$
11         end
12         Generate new children nodes of $N_j$ as
           $N_{j_1}, \dots, N_{j_{|A^*|}}$ through $A^*$
13         for $1 \leq s \leq |A^*|$ do
14             CCTree.Horizontal($N_{j_s}$)
15         end
16     end
17 end
```

The only communications between *Alice* and *Bob*, in Algorithm 4, occur at lines 4, 5 and 7, which have been proven to be secure in Theorems IV.1, IV.2, and IV.3, respectively.

## *B. Vertical Data Distribution*

Now lets us consider that *Alice* and *Bob* are interested to detect the clusters on the whole of their datasets, when data is partitioned *vertically* between two parties. This means that *Alice* and *Bob* each holds the same set of objects, but described with two different sets of attributes in each side. More precisely, let $\mathcal{A} = \{A_1, A_2, \dots, A_k\}$ be the set of categorical attributes used to describe each record of data. However, having the same set of objects, *Alice* and *Bob* each holds excluded parts of attributes describing the data. Since in reality, it is possible they also have some common attributes, we assume that the common attributes to be shared with one of them, say *Bob*. Without loss of generality, let consider *Alice* has the description of data based on $\mathcal{A}^a = \{A_1, \dots, A_{k_1}\} \subset \mathcal{A}$, and consequently *Bob* owns the set $\mathcal{A}^b = \{A_{k_1+1}, \dots, A_k\} \subset \mathcal{A}$, describing the same set of objects $D$. For instance, *Alice* may have the information about the "*color*" of objects, while *Bob* has the information about the "*size*" of the same objects.

Figure 3 depicts a higher level representation of CCTree construction when data is divided vertically between two parties. As it can be observed, both tables contain the same set of objects but expressed with different features. At the end both *Alice* and *Bob* obtain CCTree structure on whole of data information, without knowing each other data. After obtaining the final CCTree structure, it is exploited in each side for data clustering. In what follows we present secure two-party algorithms, to detect CCTree structure on vertical distributed data.

*1) Optimum Attribute Selection:* To find the optimum attribute for data division, first *Alice* and *Bob* each finds the best attribute, in terms of highest entropy, in her/his own dataset, say $A^*$ and $B^*$, respectively. Then it is required to verify if $H(A^*) > H(B^*)$. To run this verification, we plan to exploit the secure number comparison protocol presented in III-C. However, secure number comparison protocol is only applicable on natural numbers, whilst the result of attribute entropy is not necessarily a natural number. To this end, suppose all attribute entropies are rounded to maximum $l$ decimal numbers. Then, for comparing attribute entropies, it is merely enough to multiply the results by $10^l$. Algorithm 5 details the process of finding the best split attribute.

**Algorithm 5:** *Att.selection()*: Secure Attribute Selection

```
Data: Alice has $\mathcal{A}^a = \{A_1, \dots, A_{k_1}\}$, Bob has
      $\mathcal{A}^b = \{A_{k_1+1}, \dots, A_k\}$ in node $N_j$, and $l$ is the
      maximum length of decimal numbers.
Result: Alice and Bob find securely the optimum
      attribute $A^* \in \mathcal{A}^a \cup \mathcal{A}^b$ with the highest entropy.
1  initialization;
2  Alice: $A^{*a} \leftarrow argmax_{A_1 \leq A_i \leq A_{k_1}} H(A_i)$
3  Bob: $A^{*b} \leftarrow argmax_{A_{k_1+1} \leq A_i \leq A_k} H(A_i)$
4  if $SNC(10^l \cdot H(A^{*a}), 10^l \cdot H(B^{*b})) \neq 0$ then
5      return $A^{*a}$
6  else
7      return $B^{*b}$
8  end
9  return
```

**Theorem IV.4.** *Algorithm 5 only reveals the result of attribute entropy comparison.*

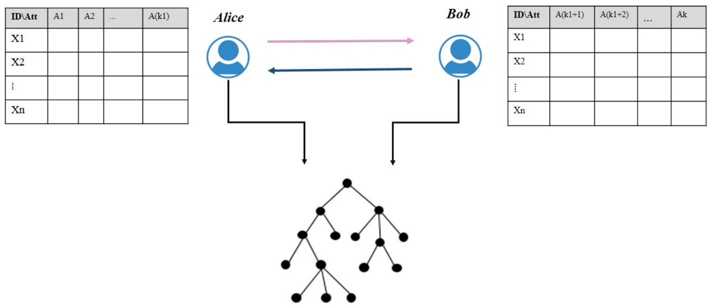

Fig. 3: Secure CCTree Construction on Vertically Partitioned Data

*Proof.* The only communication between *Alice* and *Bob* occurs at line 4, which is a call of secure number comparison protocol, proven to be secure in [20]. □

*2) Stop condition (node purity):* To verify the satisfaction of *node purity*, as a stop condition, it is simply enough that *Alice* and *Bob* compute on their own side the entropy of node $N_j$ as the following:

$$\rho(N_j^a) = \sum_{i=1}^{k_1} H(A_i) \quad , \quad \rho(N_j^b) = \sum_{i=k_1+1}^{k} H(A_i)$$

where $\rho(N_j^a)$ and $\rho(N_j^b)$ on $\mathcal{A}^a$ and $\mathcal{A}^b$ are computed with *Alice*, and *Bob*, respectively. The node purity on all attributes $\mathcal{A} = \{A_1, A_2, \ldots, A_k\}$ is obtained through:

$$\rho(N_j) = \frac{\rho(N_j^a) + \rho(N_j^b)}{k_1 + (k - k_1)}$$

This means that if secure weighted average protocol be exploited on the pairs $(\rho(N_j(A)), k_1)$ and $(\rho(N_j(B)), k - k_1)$, computed on the sides of *Alice* and *Bob* respectively, then $\rho(N_j)$ can be obtained securely. Algorithm 6 details the process.

**Algorithm 6:** *Node.purity.V()*: Secure Node Purity Computation, Vertical Distribution

```
Data: Alice has $\mathcal{A}^a = \{A_1, \ldots, A_{k_1}\}$ and Bob has
      $\mathcal{A}^b = \{A_{k_1+1}, \ldots, A_k\}$ describing data of node $N_j$.
Result: Alice and Bob securely compute
      $\rho(N_j) = \frac{1}{k}\sum_{i=1}^{k} H(A_i)$
1  initialization;
2  Alice:  $\rho(N_j^a) \leftarrow \sum_{i=1}^{k_1} H(A_i)$
3  Bob:    $\rho(N_j^b) \leftarrow \sum_{i=k_1+1}^{k} H(A_i)$
4  return $\rho(N_j) = WAP((\rho(N_j^a), k_1), (\rho(N_j^b), k - k_1))$
```

**Theorem IV.5.** *Algorithm 6 reveals nothing except the purity of a specific node.*

*Proof.* The only communication between *Alice* and *Bob* occurs at line 4, which is a call of secure weighted average protocol, proven to be secure in [19]. □

*3) Stop condition (number of elements):* In the case that two parties are interested to find that if the number of elements in a node is less than a threshold to be labeled as a leaf, first two participated parties set together this threshold, say $\mu$.

It is noticeable that when data are distributed vertically, in each new obtained node either *Alice* or *Bob* is required to declare if the condition of minimum number of elements is respected or not. The rational behind it is that the division attribute is either selected from *Alice*'s or *Bob*'s set of attributes. Hence, suppose that attribute $A$, having the highest entropy among all attributes in a node, belongs to the set of *Alice*'s attributes $\mathcal{A}^a$. Thence, after data division applying $A$, *Alice* says that whether in new obtained nodes the number of elements is less than $\mu$ or not. However, she is not required to specify the number of elements. Algorithm 7 details the process.

**Algorithm 7:** *Num.elements.V()*: Secure Number of Elements Comparison, Vertical Distribution

```
Data: Node $N_j$.
Result: Wethear $n(N_j) < \mu$ or not.
1  initialization;
2  if father of node $N_j$ is split with Alice's attribute then
3      if $n(N_j) < \mu$ then
4          Alice : $r \leftarrow 0$
5          else Alice : $r \leftarrow 1$
6      end
7  end
8  else if father of node $N_j$ is split with Bob's attribute
   then
9      if $n(N_j) < \mu$ then
10         Bob : $r \leftarrow 0$
11         else Bob : $r \leftarrow 1$
12     end
13 end
14 return $r$
```

**Theorem IV.6.** *Algorithm 7 reveals nothing about other party* *dataset information, except that if the number of elements in a node is less than specified threshold.*

*Proof.* The proof is resulted from the simple fact that no party specifies number of elements in a node and just determines if the stop condition of *number of elements in a node* satisfies or not. □

In Algorithm 8, we detail the complete process of constructing CCTree on vertical distributed data between two parties.

**Algorithm 8:** *CCTree.Vertical()*: Secure CCTree Construction, Vertical Data Distribution

```
Data: Alice holds $D$ described on $\mathcal{A}^a = \{A_1, \ldots, A_{k_1}\}$,
      Bob holds $D$ described on $\mathcal{A}^b = \{A_{k_1+1}, \ldots, A_k\}$,
      Stop conditions $\mu$, $\varepsilon$, maximum decimal length $l$.
Result: Secure (vertical) CCTree construction on $D$ and
        $\mathcal{A} = \mathcal{A}^a \cup \mathcal{A}^b$
1  initialization;
2  Root node $N_0$ contains all $D$
3  for each node $N_j \neq leaf$ do
4      if (Node.purity.V($\mathcal{A}^a, \mathcal{A}^b, N_j$) $< \varepsilon$
5      or Num.elements.V ($n(N_j^a), n(N_j^b), \mu$)) $= 0$ then
6          Label $N_j$ as leaf
7      else
8          for $1 \leq i \leq k$ do
9              $A^* \leftarrow$ Att.selection($\mathcal{A}^a, \mathcal{A}^b, N_j, l$)
10         end
11     end
12     Generate new children nodes of $N_j$ through $A^*$ as
       $N_{j_1}, \ldots, N_{j_{|A^*|}}$
13     for $1 \leq s \leq |A^*|$ do
14         CCTree.Vertical($N_{j_s}$)
15     end
16 end
```

The only communication between *Alice* and *Bob* arises at lines 5 and 9, which have been proven to be secure in Theorems IV.4, IV.5 and IV.6.

## V. Conclusion

In this work we proposed a framework which can be exploited for two parties to construct a hierarchical categorical clustering algorithm, named CCTree, on whole of their data, without revealing the original datasets. To this end, secure two-party computation algorithms are proposed to obtain the required criteria for detecting the clusters on the whole data. Two scenarios of data being distributed either horizontally or vertically have been considered. In future directions we plan to generalize the proposed framework to the wide range of clustering algorithms, where the main criteria are based on secure weighted average computation. Moreover, we plan to analyze the efficiency of proposed approach on benchmark dataset clustering to evaluate communication cost in reality.

## Acknowledgment

This work was partially supported by the H2020 EU funded project NeCS [GA #675320] and by the H2020 EU funded project C3ISP [GA #700294].

## References

[1] F. Martinelli, A. Saracino, and M. Sheikhalishahi, "Modeling privacy aware information sharing systems: A formal and general approach," in *15th IEEE International Conference on Trust, Security and Privacy in Computing and Communications*, 2016.

[2] M. Sheikhalishahi and F. Martinelli, "Privacy-utility feature selection as a privacy mechanism in collaborative data classification," in *The 26th IEEE International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises, Poznan, Poland*, 2017.

[3] E. Bogan and J. English, *Benchmarking for Best Practices: Winning Through Innovative Adaptation*, M. Hill, Ed., 1994.

[4] S. R. M. Oliveira and O. R. Zaïane, "Privacy preserving frequent itemset mining," in *Proceedings of the IEEE International Conference on Privacy, Security and Data Mining - Volume 14*, ser. CRPIT '14, 2002, pp. 43–54.

[5] M. F. Faiella, A. L. Marra, F. Martinelli, F. Mercaldo, A. Saracino, and M. Sheikhalishahi, "A distributed framework for collaborative and dynamic analysis of android malware," in *25th Conference on Parallel, Distributed, and Network-Based Processing, St. Petersburg*, 2017.

[6] C. Artoisenet, M. Roland, and M. Closon, "Health networks: actors, professional relationships, and controversies," in *Collaborative Patient Centred eHealth*, vol. 141. IOSPress, 2013.

[7] P. Berkhin, "A survey of clustering data mining techniques," in *Grouping Multidimensional Data*, J. Kogan, C. Nicholas, and M. Teboulle, Eds. Springer Berlin Heidelberg, 2006, pp. 25–71.

[8] M. Sheikhalishahi, A. Saracino, M. Mejri, N. Tawbi, and F. Martinelli, "Fast and effective clustering of spam emails based on structural similarity," in *8th International Symposium on Foundations and Practice of Security*, 2015.

[9] S. R. M. Oliveira and O. R. Zaïane, "A privacy-preserving clustering approach toward secure and effective data analysis for business collaboration," *Comput. Secur.*, vol. 26, no. 1, pp. 81–93, Feb. 2007.

[10] M. Sheikhalishahi, M. Mejri, N. Tawbi, and F. Martinelli, "Privacy-aware data sharing in a tree-based categorical clustering algorithm," in *Foundations and Practice of Security - 9th International Symposium, QC, Canada*, 2016, pp. 161–178.

[11] P. Bunn and R. Ostrovsky, "Secure two-party k-means clustering," in *Proceedings of the 14th ACM Conference on Computer and Communications Security*, ser. CCS '07. NY, USA: ACM, 2007, pp. 486–497.

[12] G. Jagannathan, K. Pillaipakkamnatt, and R. N. Wright, "A new privacy-preserving distributed k-clustering algorithm." in *SDM*. SIAM, 2006, pp. 494–498.

[13] S. Jha, L. Kruger, and P. McDaniel, *Privacy Preserving Clustering*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 397–417.

[14] A. Inan, S. V. Kaya, Y. Saygn, E. Savas, A. A. Hintoglu, and A. Levi, "Privacy preserving clustering on horizontally partitioned data," *Data and Knowledge Engineering*, vol. 63, no. 3, pp. 646 – 666, 2007, 25th International Conference on Conceptual Modeling (ER 2006).

[15] J. Vaidya, C. Clifton, M. Kantarcioglu, and A. S. Patterson, "Privacy-preserving decision trees over vertically partitioned data," *ACM Trans. Knowl. Discov. Data*, vol. 2, no. 3, pp. 14:1–14:27, Oct. 2008.

[16] M.-J. Xiao, L.-S. Huang, Y.-L. Luo, and H. Shen, "Privacy preserving id3 algorithm over horizontally partitioned data," in *Sixth International Conference on Parallel and Distributed Computing Applications and Technologies (PDCAT'05)*, Dec 2005, pp. 239–243.

[17] F. Emekci, O. Sahin, D. Agrawal, and A. E. Abbadi, "Privacy preserving decision tree learning over multiple parties," *Data and Knowledge Engineering*, vol. 63, no. 2, pp. 348 – 361, 2007.

[18] M. Sheikhalishahi, M. Mejri, and N. Tawbi, "Clustering spam emails into campaigns," in *1st International Conference on Information Systems Security and Privacy*, S. D. Library, Ed., 2015.

[19] S. Jha, L. Kruger, and P. McDaniel, *Privacy Preserving Clustering*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 397–417.

[20] A. C. Yao, "Protocols for secure computations," in *Proceedings of the 23rd Annual Symposium on Foundations of Computer Science*, ser. SFCS '82. DC, USA: IEEE Computer Society, 1982, pp. 160–164.