Scalable Distributed Data Anonymization for Large Datasets

Sabrina De Capitani di Vimercati, *Senior Member, IEEE,* Dario Facchinetti, *Member, IEEE,* Sara Foresti, *Senior Member, IEEE,* Giovanni Livraga, *Member, IEEE,* Gianluca Oldani, *Member, IEEE,* Stefano Paraboschi, *Member, IEEE,* Matthew Rossi, *Member, IEEE,* Pierangela Samarati, *Fellow, IEEE*

Abstract—k-Anonymity and ℓ-diversity are two well-known privacy metrics that guarantee protection of the respondents of a dataset by obfuscating information that can disclose their identities and sensitive information. Existing solutions for enforcing them implicitly assume to operate in a centralized scenario, since they require complete visibility over the dataset to be anonymized, and can therefore have limited applicability in anonymizing large datasets. In this paper, we propose a solution that extends Mondrian (an efficient and effective approach designed for achieving k-anonymity) for enforcing both k-anonymity and ℓ-diversity over large datasets in a distributed manner, leveraging the parallel computation of multiple workers. Our approach efficiently distributes the computation among the workers, without requiring visibility over the dataset in its entirety. Our data partitioning limits the need for workers to exchange data, so that each worker can independently anonymize a portion of the dataset. We implemented our approach providing parallel execution on a dynamically chosen number of workers. The experimental evaluation shows that our solution provides scalability, while not affecting the quality of the resulting anonymization.

Index Terms—Distributed data anonymization, Mondrian, k-Anonymity, ℓ-Diversity, Apache Spark

1 Introduction

Guaranteeing privacy in datasets containing possible identifying and sensitive information requires not only refraining from publishing explicit identities, but also obfuscating data that can leak (disclose or reduce uncertainty of) such identities as well as their association with sensitive information. k-Anonymity [3], [4], extended with ℓ-diversity [5], offers such protection. k-Anonymity requires generalizing values of the *quasi-identifier* attributes (i.e., attributes that can expose to linkage with external sources and leak information on respondents' identities) to ensure each quasi-identifier combination of values to appear at least k times. ℓ-Diversity considers each sensitive attribute in grouping tuples for quasi-identifier generalization so to ensure each group of tuples (whose quasi-identifiers will then be generalized to the same values) be associated with at least ℓ different values of the sensitive attribute.

While simple to express, k-anonymity and ℓ-diversity are far from simple to enforce, given the need to balance privacy (in terms of the desired k and ℓ) and utility (in terms of information loss due to generalization). Also, the computation of an optimal solution requires evaluating (based on the dataset content) which quasi-identifying attributes generalize and how, and hence demands complete visibility of the whole dataset. Hence, existing solutions implicitly assume to operate in a centralized environment. Such an assumption clearly does not fit large scale systems where the amount of data collected is huge (there are widely circulating estimates that a smart car uploads to the cloud 25GB per hour). While scalable distributed architectures can help in performing computation on such large datasets, their use in computing an optimal k-anonymous solution requires careful design. In fact, a simple distribution of the anonymization load among workers would affect either the quality of the solution or the scalability of the computation (requiring expensive synchronization and data exchange among workers [6]).

In this paper, we address the problem of efficiently anonymizing large data collections. We propose a solution that extends Mondrian [7], an efficient and effective approach originally proposed for achieving k-anonymity in a centralized scenario, to enforce both k-anonymity and ℓ-diversity in a distributed scenario. With our approach, anonymization is executed in parallel by multiple workers, each operating on a portion of the original dataset to be anonymized. The design of our partitioning approach aims at limiting the need for workers to exchange data, by splitting the dataset to be anonymized into as many partitions as the number of available workers, which can independently run our revised version of Mondrian on their portion of the data. A distinctive feature of our proposal is that the partitioning approach does not require knowledge of the entire dataset to be anonymized. Rather, it can be executed on a sample of the dataset whose size can be dynamically

- *S. De Capitani di Vimercati, S. Foresti, G. Livraga, and P. Samarati are with the Università degli Studi di Milano, Italy.*
  E-mail: firstname.lastname@*unimi.it*
- *D. Facchinetti, G. Oldani, S. Paraboschi, and M. Rossi are with the Università degli Studi di Bergamo, Italy.*
  E-mail: firstname.lastname@*unibg.it*

Preliminary versions of this paper appeared under the titles "Scalable Distributed Data Anonymization," in Proc. of the 19th IEEE International Conference on Pervasive Computing and Communications (PerCom 2021) *[1] and "Artifact: Scalable Distributed Data Anonymization," in* Proc. of the 19th IEEE International Conference on Pervasive Computing and Communications (PerCom 2021) *[2]*

adjusted. The approach is therefore applicable in scenarios where the dataset is very large, maybe even distributed, and does not entirely fit in main memory. We have implemented our approach providing parallel execution on a dynamically chosen number of workers. The experimental evaluation confirms both the goodness of our partitioning strategy with respect to maintaining utility of the anonymized dataset and the scalability of our approach. The main contributions of this paper can be summarized as follows. First, we propose and evaluate different partitioning strategies for distributing data to workers. Second, we extend the original Mondrian algorithm to operate in a distributed scenario without asking workers to interact, and to enforce both k-anonymity and ℓ-diversity. Third, we support different strategies for managing generalization, including the use of generalization hierarchies that permit to produce semantically-aware anonymization. Fourth, we evaluate different metrics for assessing the information loss caused by the distribution of the anonymization process.

The remainder of this paper is organized as follows. Section 2 discusses basic concepts over which our solution builds. Section 3 presents an overview of our approach for distributed anonymization, modeling the reference scenario and illustrating the actors involved, and a sketch of our anonymization approach. Section 4 discusses our approach for partitioning the dataset to be anonymized in fragments to be assigned to the workers for anonymization. Section 5 illustrates how workers can independently anonymize the fragments allocated to them. Section 6 describes how information loss is assessed. Section 7 presents the implementation of our distributed anonymization algorithm. Section 8 illustrates experimental results. Section 9 discusses related work. Finally, Section 10 concludes the paper.

2 Basic concepts

Our solution is based on three main pillars: k-anonymity, ℓ-diversity, and Mondrian.

k-Anonymity. k-Anonymity [3] is a privacy property aimed at protecting respondents identities in data publication. k-Anonymity starts from the observation that a dataset, even if de-identified (i.e., with explicit identifying information removed) can contain other attributes, called *quasi-identifiers* (abbreviated QI) such as gender, date of birth, and living area, that can be exploited for linking the dataset with other data sources and enable observers to reduce uncertainty on the identity (or identities) to whom the tuples in the de-identified dataset refer. k-Anonymity demands that no tuple in a released dataset can be related to less than a certain number k of respondents. k-Anonymity operates on the values of the QI attributes to ensure that no tuple can be uniquely associated with the identity of its respondent through its QI values, and vice versa. In practice, k-anonymity is enforced by ensuring (through generalization of data values) that each combination of values of the quasi-identifier in a dataset appears with at least k occurrences. In this way, any linking attack exploiting the quasi-identifier will always find at least k individuals to which each anonymized tuple can correspond and vice versa. k-Anonymity can be guaranteed in different ways. The original proposal of k-anonymity applies generalization

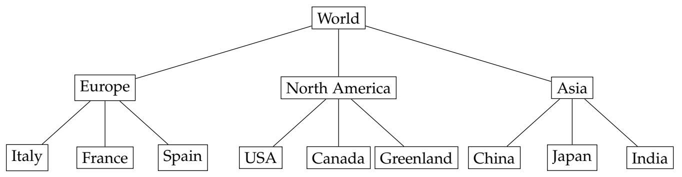
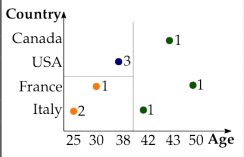


Fig. 1: Generalization hierarchy for attribute `Country`

Age	Country	TopSpeed
25	Italy	132
25	Italy	132
30	France	128
42	Italy	110
50	France	115
43	Canada	115
38	USA	126
38	USA	127
38	USA	140

(a)



(b)

Age	Country	TopSpeed
[25,30]	Europe	132
[25,30]	Europe	132
[25,30]	Europe	128
[42,50]	World	110
[42,50]	World	115
[42,50]	World	115
38	USA	126
38	USA	127
38	USA	140

(c)

Fig. 2: An example of a dataset (a), its spatial representation and partitioning (b), and a 3-anonymous and 2-diverse version (c), considering quasi-identifier QI={`Age`,`Country`} and sensitive attribute `TopSpeed`

to the QI attributes [3]. Generalization is a data protection technique that replaces attribute values with other, more general values. For instance, an individual's `Age` may be generalized in age ranges (e.g., replacing all age values from 25 to 30 with a single interval $[25, 30]$). While numeric attributes (i.e., attributes defined on a totally ordered domain) naturally generalize to ranges of values, the generalization of categorical attributes (i.e., attributes defined on a non-ordered domain) can leverage generalization hierarchies (e.g., Figure 1 illustrates a generalization hierarchy for attribute `Country`). Since generalization (while maintaining data truthfulness) removes details from data, it reduces the risk of finding unique correspondences for QI values with external data sources. For example, the dataset in Figure 2(c) is a 3-anonymous version of the dataset in Figure 2(a), considering attributes `Age` and `Country` as quasi-identifer. In the figure, quasi-identifying attributes `Age` and `Country` have been generalized so that their values appear with at least 3 occurrences (for readability, the i^{th} tuple in Figure 2(a) corresponds to the i^{th} tuple in Figure 2(c)). For example, the `Age` and `Country` of the respondents of the first three tuples have been generalized to range $[25, 30]$ and to value *Europe*, respectively. Note that neither the values for `Age` nor the values for `Country` of the last three tuples have been generalized to obtain 3-anonymity, since they already share the same values for the quasi-identifier (38 and *USA*, respectively). It is easy to see that no record in external data sources can be linked, through `Age` and `Country`, to less than three tuples in the 3-anonymous dataset.

ℓ-Diversity. ℓ-Diversity [5] extends k-anonymity to prevent attribute disclosure, that is, to protect against possible inferences aimed at associating a value for the sensitive attribute to the respondent's identity. With reference to the datasets in Figure 2, suppose that the `TopSpeed` of the respondent aged 30 from *France* (third tuple) were 132. The 3-anonymous dataset in Figure 2(c) would have, for the first three tuples, the same sensitive value (132). While, thanks to the protection offered by 3-anonymity, no record in an external data source (e.g., a voter list) can be uniquely mapped to any of these tuples, this 3-anonymous dataset would still leak the fact that European respondents with `Age` between 25 and 30 have `TopSpeed` equal to 132. ℓ-Diversity extends k-anonymity by demanding that each equivalence class E (i.e., each set of tuples sharing the same generalized values for the quasi-identifier) have at least ℓ well-represented values for the sensitive attribute(s). Several definitions of *well-represented* have been proposed, and a natural interpretation requires at least ℓ different values for the sensitive attribute(s). For example, the 3-anonymous dataset in Figure 2(c) is also 2-diverse, since each equivalence class contains at least two different values for `TopSpeed`.

Mondrian. Mondrian [7] is a multi-dimensional algorithm that provides an efficient and effective approach for achieving k-anonymity. Mondrian leverages a spatial representation of the data, mapping each quasi-identifier attribute to a dimension and each combination of values of the quasi-identifier attributes to a point in such a space. Mondrian operates a recursive process to partition the space in regions containing a certain number of points (which corresponds to splitting the dataset represented by the points in the space in fragments that contain a certain number of records). In particular, at each iteration Mondrian cuts the set of tuples in each fragment F computed at the previous iteration (the whole dataset at the first step) based on the values (e.g., for numerical attributes, whether lower/higher than the median) for a quasi-identifying attribute chosen for each cut. The algorithm terminates when any further cut would generate only sub-fragments with less than k tuples, at which point values of the quasi-identifying attributes in each fragment are substituted with their generalization. Figure 2(b) shows the spatial representation and partitioning of the dataset in Figure 2(a), where the number associated with each data point is the number of tuples with such values for quasi-identifier `Age` and `Country` in the dataset. The 3-anonymous version of the dataset in Figure 2(c) has been obtained by first partitioning the dataset in Figure 2(a) based on attribute `Age`: fragment $F_{\texttt{Age}\leq 38}$ includes all the tuples with `Age` less than or equal to the median value 38, and $F_{\texttt{Age}>38}$ the remaining tuples. $F_{\texttt{Age}\leq 38}$ is further partitioned based on attribute `Country`, obtaining $F_{\texttt{Age}\leq 38,\ \texttt{Country}\ \text{IN}\ \{\textit{Canada,USA}\}}$ including all tuples with `Country` equal to *Canada* or *USA*, and $F_{\texttt{Age}\leq 38,\ \texttt{Country}\ \text{IN}\ \{\textit{France,Italy}\}}$ including the remaining tuples. No further partitioning is possible (all fragments include exactly three tuples), and the quasi-identifying attributes in each fragment can be generalized. The dataset in Figure 2(c) has been obtained generalizing the dataset in Figure 2(a) according to the partitioning in Figure 2(b) and leveraging the generalization hierarchy in Figure 1 for attribute `Country`.

3 Distributed anonymization

We consider a scenario where a large and maybe distributed dataset $\mathcal{D}$ needs to be anonymized, and may not entirely fit into the main memory of a single machine. Our goal is then to distribute the anonymization of $\mathcal{D}$ to a set $W = \{w_1, \ldots, w_n\}$ of workers so that they can operate in parallel and independently from one another, to have

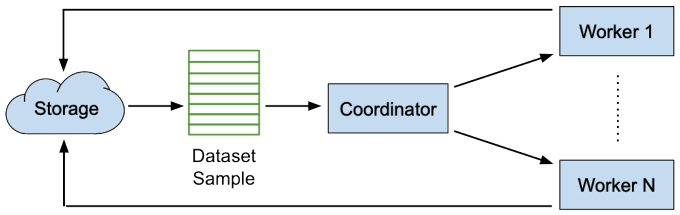


Fig. 3: Overall view of the distributed anonymization process

benefits in terms of performance while not compromising on the quality of the solution (with respect to a traditional centralized anonymization of $\mathcal{D}$). To this purpose, we extend Mondrian to operate in such a way that workers in W are assigned (non-overlapping) partitions of $\mathcal{D}$ (i.e., sets of tuples of $\mathcal{D}$, which we call *fragments*) and can operate limiting the need for data exchanges with other workers. Each worker $w \in W$ can independently anonymize its fragment satisfying k-anonymity and ℓ-diversity, with the guarantee that the combination of the anonymized fragments is a k-anonymous and ℓ-diverse version of $\mathcal{D}$. The overall process is overseen by a Coordinator, and includes a *pre-processing phase* (which partitions the dataset $\mathcal{D}$ in fragments and assigns fragments to workers) and a *wrap-up phase* (which collects the anonymized fragments from the workers, recombines them, and evaluates the quality of the computed solution). Our reference scenario (as graphically represented in Figure 3) is then characterized by a *(distributed) storage platform, storing and managing the dataset $\mathcal{D}$ to be anonymized (as well as its anonymized version $\hat{\mathcal{D}}$*, after workers have anonymized their fragments), the anonymizing *workers*, and the Coordinator. In the remainder of this paper, given a dataset $\mathcal{D}$ with quasi-identifier QI=$\{a_1, \ldots, a_q\}$ and privacy parameters k and ℓ, we denote with $\hat{\mathcal{D}}$ the k-anonymous and ℓ-diverse version of $\mathcal{D}$; with $\hat{t}\in\hat{\mathcal{D}}$ the generalized version of t in the anonymized dataset, $\forall t\in\mathcal{D}$; and with $\hat{F}$ the anonymized version of fragment F (i.e., $\forall t \in F$, $\exists \hat{t} \in \hat{F}$ s.t. $\hat{t}$ is the generalized version of t).

The pre-processing phase is crucial for our distributed anonymization. The first problem to be addressed is the definition, by the Coordinator, of a fragmentation strategy, regulating which tuples belong to which fragment. An effective strategy, as demonstrated by our experimental results (Section 8), is to fragment $\mathcal{D}$ based on the values of (some of the) quasi-identifying attributes $a_1, \ldots, a_h$, in such a way that a tuple t of $\mathcal{D}$ is assigned to a fragment F based on the values of $t[a_1], \ldots, t[a_h]$. To illustrate, consider the dataset in Figure 2(a) and suppose to define two fragments F_1 and F_2 based on the values of `Age`. The Coordinator may define a strategy such that F_1 contains all tuples of $\mathcal{D}$ with values lower than or equal to 38 (i.e., the first three and last three tuples of the table), and F_2 the remaining tuples of $\mathcal{D}$. In principle, this would require the Coordinator to have complete visibility over $\mathcal{D}$ for defining fragments. However, $\mathcal{D}$ might be too large to fit into the main memory of the Coordinator. We then propose a strategy in which the Coordinator can define the conditions that regulate the fragmentation of $\mathcal{D}$ based on a *sample* of $\mathcal{D}$, whose size can be dynamically adjusted according to the storage capabilities of the Coordinator. The Coordinator then communicates the conditions to the workers, which will then download the tuples in $\mathcal{D}$ that satisfy such conditions directly from the storage platform (i.e., without the need for the Coordinator to send any dataset to the workers). With reference to the example above, where fragments are defined based on the values of `Age`, the Coordinator communicates to workers the value ranges (e.g., lower than or equal to 38, and greater than 38) for the tuples in their fragments. The anonymization phase following the pre-processing operates in parallel at the workers. For the design of this phase, we specifically focus on the support, for categorical attributes, of generalization hierarchies to the aim of producing semantically-aware generalized (anonymous) data. Our pre-processing phase is discussed in Section 4, while the anonymization and wrap-up phases are described in Sections 5 and 6, respectively.

4 Data pre-processing

The pre-processing phase of our approach operates on a sample D of $\mathcal{D}$, whose size is tuned depending on the storage capabilities of the Coordinator. For the sake of readability, we refer our discussion to a generic dataset D with the note that D is a sample of the original dataset $\mathcal{D}$ to be anonymized (clearly the quasi-identifier attributes considered for D are those defined for $\mathcal{D}$).

4.1 Partitioning strategies

Selecting a strategy for partitioning the dataset D is crucial in our scenario, since a random partitioning may cause considerable information loss. Indeed, if fragments include tuples with heterogeneous values for the quasi-identifier, each worker (which independently operates on its fragment) would need a considerable amount of generalization to satisfy k-anonymity. On the contrary, information loss is mitigated if partitioning does not spread across fragments tuples that assume similar values for the quasi-identifier. To illustrate, consider a dataset with four tuples $t_1, \ldots, t_4$ having values 25, 25, 60, and 60 for `Age`, which needs to be partitioned in two fragments. If partitioning generates two fragments $F_1=\{t_1,t_2\}$ and $F_2=\{t_3,t_4\}$, no generalization is needed to enforce 2-anonymity. On the contrary, fragments $F_1=\{t_1,t_3\}$ and $F_2=\{t_2,t_4\}$ requires generalizing `Age` to the range [25,60] in each fragment, causing higher information loss.

To limit the information loss implied by the partitioning of a dataset D with quasi-identifier QI among a set $W=\{w_1,\ldots,w_n\}$ of workers, we propose two strategies.

- The *quantile-based approach* selects an attribute a from the QI, and partitions D in n fragments $F_1,\ldots,F_n$ according to the n-quantiles of a in D.
- The *multi-dimensional approach* recursively partitions D in a similar way as the Mondrian approach (see Section 2). Given a fragment F (the entire dataset D at the first iteration), the multi-dimensional strategy selects an attribute $a\in$QI and partitions F in two fragments according to the median value of a in F. Each of the resulting fragments is then further partitioned, until n fragments have been obtained.

```
Q_PARTITION($D, W$)
1: let $a$ be the attribute used to partition $D$
2: $R := \{rank(t[a]) \mid t\in D\}$ /* rank of $a$'s values in the ordering */
3: let $q_i$ be the $i^{th}$ $|W|$-quantile for $R, \forall i = 1,\ldots,|W|$
4: $F_1 := \{t\in D \mid rank(t[a]) \leq q_1\}$
5: for each $i = 2,\ldots,|W|$ do
6:     $F_i := \{t\in D \mid q_{i-1} < rank(t[a]) \leq q_i\}$
```

Fig. 4: Quantile-based partitioning

Both the quantile-based and the multi-dimensional partitioning approaches rely on an ordering among the values that the attribute a selected for partitioning assumes in D to compute quantiles (for the quantile-based approach) and median values (for the multi-dimensional approach). In fact, given a sample D of the dataset and the attribute a selected for partitioning, the tuples in D are first ordered according to their value of a, establishing a ranking among the attribute values. Quantiles and the median values are then computed on such a ranking. When a is numerical, ordering among values is naturally defined. When a is categorical and has a generalization hierarchy $\mathcal{H}(a)$, attribute values are considered with the order in which they appear in the leaves of $\mathcal{H}(a)$, aiming at keeping in the same fragment values that generalize to a more specific value. Indeed, leaf values that are close in the hierarchy will have a common ancestor (to which they would be generalized) at a lower level in the hierarchy (see Section 5), thus limiting information loss. For instance, with reference to the hierarchy in Figure 1, we use order $\langle$*Italy*, *France*, *Spain*, *USA*, *Canada*, *Greenland*, *China*, *Japan*, *India*$\rangle$. This ordering would combine in the same fragment values *Italy* and *France*, which generalize to a more specific value (i.e., *Europe*) than a fragment with values *Italy* and *Canada* (i.e., *World*).

Figure 4 illustrates the procedure implementing quantile-based partitioning executed by the Coordinator. Given a dataset D and a set W of workers, the procedure selects the attribute a for partitioning, orders the tuples in D according to $t[a]$, and determines the rank $rank(t[a])$ of each value $t[a]$ (lines 1–2). It then computes the $|W|$-quantiles of such ranking R (line 3). The first fragment F_1 is obtained by including all the tuples $t\in D$ with rank of $t[a]$ lower than or equal to the first computed quantile (line 4). The remaining fragments $F_2,\ldots,F_{|W|}$ are obtained by including in F_i all the tuples $t\in D$ with ranks of $t[a]$ in the interval $(q_{i-1},q_i]$, with q_i the i^{th} $|W|$-quantile of the computed ranks (lines 5–6). To illustrate, consider partitioning in 4 fragments the dataset in Figure 2(a) with the quantile-based approach over attribute `Age` (for simplicity, we consider the dataset in Figure 2(a) as a sample). For the first tuple t_1, we have that $rank(t_1[\texttt{Age}])=rank(25)=1$, since 25 is the first value (i.e., the smallest) in the ordering for `Age`. Similarly, for the third tuple t_3 we have that $rank(t_3[\texttt{Age}])=rank(30)=2$, since 30 is the second value in the ordering for `Age`. The 4-quantiles $q_1,\ldots,q_4$ for such ranks are q_1=2, q_2=3, q_3=4, and q_4=6. The first fragment F_1 then includes all the tuples t such that $rank(t[\texttt{Age}])\leq 2$, that is, all the tuples such that $t[\texttt{Age}]\leq 30$. Fragment F_2 includes all tuples t such that

```
M_PARTITION($D, W, i$)
1: let $a$ be the attribute used to partition $D$
2: $R := \{rank(t[a]) \mid t \in D\}$ /* rank of $a$'s values in the ordering */
3: let $m$ be the median of $R$
4: $F_1 := \{t \in D \mid rank(t[a]) \leq m\}$
5: $F_2 := \{t \in D \mid rank(t[a]) > m\}$
6: if $i < \lceil \log_2 |W| \rceil$ then
7:     M_Partition($F_1, W, i+1$)
8:     M_Partition($F_2, W, i+1$)
```

Fig. 5: Multi-dimensional partitioning

2<$rank(t[\texttt{Age}])$≤3. Fragments F_3 and F_4 are computed in a similar way.

Figure 5 illustrates the recursive procedure implementing multi-dimensional partitioning executed by the Coordinator. The procedure takes as input a dataset D, the set W of workers, and the recursive level of iteration i (1 at the first invocation). The procedure first selects the attribute a for partitioning, orders the tuples in D according to $t[a]$, and determines the rank $rank(t[a])$ of each value $t[a]$ (lines 1–2). It then computes the median value m for R (line 3), and defines two fragments F_1, including the tuples t of D having rank for $t[a]$ lower than or equal to m, and F_2, including the remaining tuples (lines 4–5). The procedure then recursively calls itself on the two computed fragments (lines 7–8) with $i+1$ to further fragment them, unless the necessary number of iterations have already been executed (i.e., $i = \lceil \log_2 |W| \rceil$, since at each iteration the number of fragments doubles and $\lceil \log_2 |W| \rceil$ fragments are sufficient to assign at least a fragment to each worker) (line 6). To illustrate, consider partitioning in 4 fragments the (sample) dataset in Figure 2(a) with the multi-dimensional approach, and suppose that at the first iteration (i=1) the attribute chosen for partitioning is Age. The median of the ranks for the values of Age is 3, and the sample is split in two fragments F_{12} and F_{34} such that F_{12} includes all tuples t for which $rank(t[\texttt{Age}])$≤3, and F_{34} the remaining ones. At the second (and last) iteration (i=2), the procedure selects an attribute (which could possibly be different from Age) for fragment F_{12} and an attribute for F_{34}, and partitions each fragment in two more fragments based on the median of the ranks of such attribute values in F_{12} (F_{34}, respectively). Since 4 fragments have been obtained, the partitioning process terminates.

The quantile-based and the multi-dimensional partitioning exhibit different behavior in the definition of the fragments. The quantile-based partitioning ensures balancing among the fragments, since all n fragments will include (approximately) the same number of tuples, but its application can be limited by the domain of the attribute a chosen for partitioning (it cannot be used if the number n of workers is larger than the domain of a). On the contrary, while being always applicable, the multi-dimensional approach may result in some workers being assigned twice the workload of other workers. Since multi-dimensional partitioning doubles the number of fragments at each iteration, when the number n of workers is a power of 2, the recursive process can be executed $\log_2 n$ times, obtaining n fragments of (approximately) the same size. However, n may not be a power of 2. Aiming at using all the workers, the partitioning strategy stops when $n \leq 2^i$ (multi-dimensional partitioning generates 2^i fragments at the i-th iteration) and $2^i - n$ workers are assigned two fragments, resulting in some workers having twice the workload of the others. For instance, assume $W = \{w_1, \ldots, w_7\}$ and $|D| = 1000$. Multi-dimensional partitioning needs 3 iterations for generating at least 7 fragments ($2^2 < 7 \leq 2^3$). Since $2^i - n = 8 - 7 = 1$, one worker (e.g., w_1) will be assigned two fragments, resulting in a workload of nearly 250 tuples for w_1 and of 125 tuples for each of the other workers.

The computational complexity of the two approaches is slightly different, with quantile-based partitioning resulting more efficient than multi-dimensional partitioning (as confirmed by the experimental results in Section 8). Procedure **Q_Partition** costs $O(|D|)$, while procedure **M_Partition** costs $O(|D| \log |W|)$. The first two steps (lines 1-2) are the same for the two procedures and cost $O(|D|)$, and the computation of quantiles has the same cost as the computation of the median and cost $O(|D|)$. The **for each** loop at line 5 of procedure **Q_Partition** has cost $O(|W|)$, and therefore the overall cost of quantile-based partitioning is $O(|D|)$, since the number of workers is smaller than the number of tuples in the dataset. The recursive calls of procedure **M_Partition** imply a cost of $O(|D| \log |W|)$ since the procedure recursively calls itself with i from 1 to $\lceil \log_2 |W| \rceil$ and, for each value of i, the overall size of the fragments input to the different recursive calls is $|D|$.

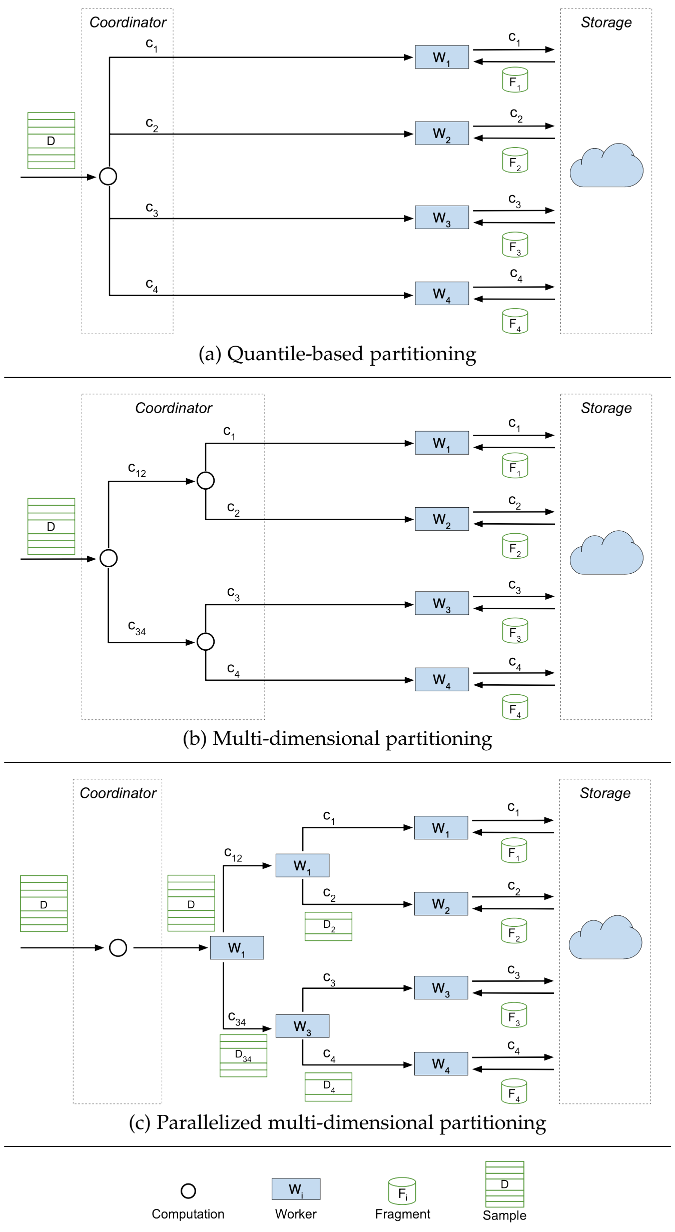
4.2 Fragments retrieval

While the Coordinator operates on a sample D of the dataset $\mathcal{D}$ to be anonymized for defining fragments, workers need to operate on the whole fragment assigned to them. To minimize communication overhead, our approach defines fragments assigned to workers according to the *partitioning conditions* identified by the Coordinator to partition D. These conditions, comparing the attribute a selected for partitioning with the values in its domain corresponding to the quantiles or median value of the ranking of tuples in D according to a, are communicated to workers (see Section 7). Each worker can then retrieve the tuples in its fragment directly, without need for the Coordinator to retrieve and communicate such tuples. For instance, consider two fragments F_1 and F_2 computed over a sample D of $\mathcal{D}$ with the multi-dimensional approach, and assume to adopt attribute Age for partitioning and that the median of the ranking corresponds to value 38 in the attribute domain. The worker in charge of the anonymization of F_1 will retrieve all the tuples in $\mathcal{D}$ having Age≤38, while the worker in charge of the anonymization of F_2 will retrieve all the tuples in $\mathcal{D}$ having Age>38.

Given a set W=$\{w_1, \ldots, w_n\}$ of workers, c_i denotes the condition describing the fragment F_i assigned to w_i, $i = 1, \ldots, n$. When using the quantile-based approach, condition c_i, $i = 1, \ldots, n$, describes the values for a that are included in the i-th n-quantile of D (i.e., its endpoints). For example, with reference to the example in Section 4.1 for the quantile-based partitioning of the dataset in Figure 2(a) in 4 fragments over attribute Age, the conditions identifying fragments $F_1, \ldots, F_4$ would be defined as c_1="(Age≤30)"; c_2="(Age>30) AND (Age≤38)";

c_3="(Age>38) AND (Age≤42)"; and c_4="(Age>42) AND (Age≤50)". When using the multi-dimensional approach, condition c_i, $i = 1, \ldots, n$, is a conjunction of conditions of the form $a \leq v$ or $a > v$ describing the recursive partitioning performed by the Coordinator to obtain fragment F_i. Intuitively, each recursive call to **M_Partition** (Figure 5) partitions the input fragment D into two fragments F_1 and F_2, described by condition c AND $(a \leq v)$ or c AND $(a > v)$, respectively, with c the condition describing the input fragment D (empty at the first iteration), a the attribute selected for partitioning, and v the value in the domain of a corresponding to the median m of the ranking of the tuples in D according to a (i.e., v=$t[a]$ s.t. $rank(t[a])$=m). To illustrate, consider the example in Section 4.1 for the multi-dimensional partitioning of the dataset in Figure 2(a) in 4 fragments. The first partitioning, based on attribute Age, produces F_{12} and F_{34} described by conditions Age≤38 and Age>38, respectively. At the second iteration, assume that F_{12} is split into fragments F_1 and F_2 according to the value of attribute Country, which is *Italy* or *France* in F_1 and *USA* or *Canada* in F_2. The conditions describing fragments F_1 and F_2 (and communicated to the corresponding workers) would be c_1 = "(Age≤38) AND (Country IN {*Italy*, *France*})"; c_2 = "(Age≤38) AND (Country IN {*USA*, *Canada*})". Figures 6(a)–(b) graphically illustrate the pre-processing operated by the Coordinator to partition a sample D of $\mathcal{D}$ assuming to produce 4 fragments (to be then assigned to 4 workers) adopting quantile-based and multi-dimensional partitioning. In the figure, we denote with $c_{i\ldots j}$ the condition describing the fragment of D that will be further partitioned to generate $F_i, \ldots, F_j$.

We close this section with a note on the possibility of leveraging the availability of multiple workers for the pre-processing. The multi-dimensional approach (Figures 5 and 6(b)) could in fact be performed in parallel by workers (see Figure 6(c)). Intuitively, each of the two fragments F_1 and F_2 produced by the partitioning of a fragment F (D at the first iteration) can be assigned to two different workers for further partitioning, so that partitioning can run in parallel (i.e., one fragment can be partitioned by the same worker in charge of splitting F while the other can be delegated to a different worker). We investigated this strategy, both theoretically and experimentally and, while clearly permitting to reduce the computation effort for the Coordinator, it would require data transfer among workers, resulting in lower performance than the traditional (non parallelized) multi-dimensional partitioning.



(a) Quantile-based partitioning

(b) Multi-dimensional partitioning

(c) Parallelized multi-dimensional partitioning

Fig. 6: Partitioning strategies

4.3 Attributes for partitioning

The first step for partitioning the dataset, regardless of the approach (i.e., quantile-based or multi-dimensional) adopted, is the selection of the quasi-identifying attribute a used to split (line 1 in procedures **Q_Partition** in Figure 4 and **M_Partition** in Figure 5).

For quantile-based partitioning, we select the attribute $a_i \in$QI with more distinct values in D. This strategy distributes the values for a_i among different fragments, limiting the necessary amount of generalization over a_i. Indeed, generalization needs to operate only over the subset of values for a_i appearing in the fragment, which are expected to be close. On the contrary, partitioning according to a different attribute a_j having a limited number of values might cause excessive generalization for a_i, if a fragment has tuples with values at the extremes of the domain for the attribute.

For multi-dimensional partitioning, similarly to the original Mondrian approach, we select the attribute $a \in$QI that has, in the fragment F to be partitioned, the highest representativity of the values it assumed in D. If a is numerical, its representativity is defined as the ratio between the span (i.e., the width of the range) of the values in F and the span of the values in the entire dataset D. If a is categorical, its representativity can be defined as the ratio between the number of distinct values in F and the number of distinct values in D. Formally, the representativity $rep(a)$ of a quasi-identifying attribute $a \in$ QI is defined as follows:

$$rep(a) = \begin{cases} \frac{\max_F\{t[a]\}-\min_F\{t[a]\}}{\max_D\{t[a]\}-\min_D\{t[a]\}} & \text{if } a \text{ is numerical} \\ \frac{\text{count}_F(\text{distinct } t[a])}{\text{count}_D(\text{distinct } t[a])} & \text{if } a \text{ is categorical} \end{cases} \qquad (1)$$

where $\max_F\{t[a]\}$ and $\min_F\{t[a]\}$ ($\max_D\{t[a]\}$ and $\min_D\{t[a]\}$, resp.) are the maximum and minimum values for a assumed by the tuples in fragment F (in dataset D, resp.); and $\text{count}_F(\text{distinct } t[a])$ ($\text{count}_D(\text{distinct } t[a])$, resp.), is the number of distinct values assumed by attribute a in F (in D, resp.). For both numerical and categorical attributes, $rep(a) \in (0, 1]$. For example, consider a fragment F with QI $= \{a_1, a_2, a_3\}$, where a_1 is categorical and a_2 and a_3 are numerical. Suppose that: *i)* the distinct values for a_1 are 1000 in D and 100 in F; *ii)* the values for a_2 are in a range of width 1000 in D and of width 500 in F; and *iii)* the values of a_3 are in a range of width 1000 in D and of width 700 in F. Since $rep(a_1) = 100/1000 = 0.1 < rep(a_2) = 500/1000 = 0.5 < rep(a_3) = 700/1000 = 0.7$, a_3 is chosen for partitioning F.

Note that, at the first iteration of multi-dimensional partitioning, all quasi-identifying attributes have representativity equal to 1 because F=D (the entire dataset D is to be partitioned). In these cases, we select the attribute with the maximum number of distinct values in the dataset (like in quantile-based partitioning).

5 Data anonymization

During the anonymization phase of our approach, each worker anonymizes the fragment assigned to it, independently (i.e., without interacting with other workers) executing our distributed version of the Mondrian algorithm. In particular, our version of the algorithm enforces also ℓ-diversity, besides k-anonymity considered by the original approach [7]. To this end, the recursive partitioning at the core of the anonymization algorithm (Section 2) terminates when any further sub-partitioning would generate fragments that have less than k occurrences for the combination of values for the quasi-identifying attributes (which would violate k-anonymity), or less than ℓ values for the sensitive attributes (which would violate ℓ-diversity). The anonymization phase of our distributed Mondrian has computational complexity $O(|F|\log|F|)$, with F the fragment input to function **Anonymize** in Figure 7. Indeed, the cost of lines 1–8 is $O(|F|)$, as discussed in Section 4. Due to the recursive calls on a partition of F including two fragments of size $\frac{|F|}{2}$, the overall complexity is $O(|F|\log|F|)$, which is in line with the complexity of the original Mondrian algorithm [7].

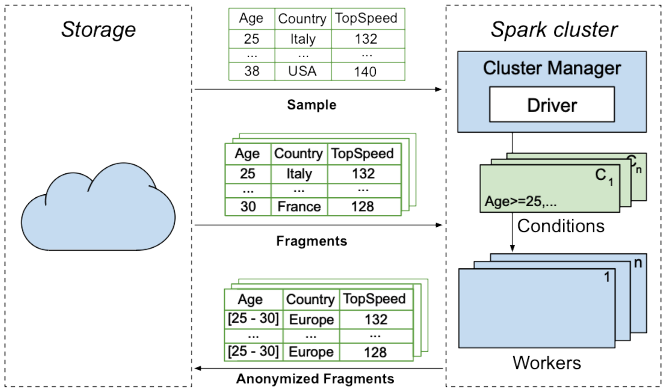
Figure 7 illustrates the anonymization algorithm executed by each worker for anonymizing the fragment assigned to it. The recursive partitioning (lines 3–10), which operates according to the same logic as multi-dimensional partitioning in the pre-processings phase (Section 4), terminates when any further sub-partitioning of a fragment would violate k-anonymity or ℓ-diversity (lines 1–2). The attribute a with maximum representativity (Equation 1) is chosen for partitioning (line 4). Clearly, since during the anonymization phase each worker w_i has visibility only on its fragment F_i, representativity is computed over F_i (in contrast to the entire dataset D in Equation 1). Like in multi-dimensional partitioning, at the first recursive call, representativity is equal to 1 for all quasi-identifying attributes. Our approach then selects the attribute that has the highest number of distinct values in the fragment. When a fragment F cannot be further partitioned as this would violate k-anonymity or ℓ-diversity, the anonymization algorithm produces the anonymized version of F, obtained generalizing the values of the quasi-identifying attributes to guarantee that all the tuples share the same (generalized) quasi-identifier values (line 2). Our distributed Mondrian approach supports the following generalization strategies.

```
ANONYMIZE($F$)
 1: if no partitioning can be done without violating
    $k$-anonymity or $\ell$-diversity then
 2:     generalize $F$[QI]
 3: else
 4:     let $a$ be the attribute for partitioning
 5:     $R := \{rank(t[a]) \mid t \in F\}$ /* rank of $a$'s values in the ordering */
 6:     let $m$ be the median of $R$
 7:     $F_1 := \{t \in F \mid rank(t[a]) \leq m\}$
 8:     $F_2 := \{t \in F \mid rank(t[a]) > m\}$
 9:     Anonymize($F_1$)
10:     Anonymize($F_2$)
```

Fig. 7: Anonymization algorithm for a fragment F

- *Generalization hierarchies*: applicable to categorical attributes only, the values for attribute $a \in$QI are substituted with their lowest common ancestor in the generalization hierarchy $\mathcal{H}(a)$ defined for a (Section 3). To illustrate, consider the dataset in Figure 2(a) and suppose, for simplicity, it is a fragment retrieved by a worker for anonymization. Its anonymized version in Figure 2(c) is obtained generalizing attribute `Country` according to the generalization hierarchy in Figure 1, considering the partitions in Figure 2(b). For example, values *Italy*, *Italy*, and *France* of the first three tuples (partition at the bottom-left of Figure 2(b)) are generalized to their lowest common ancestor in the hierarchy (i.e., *Europe*).
- *Common prefix:* applicable to categorical and numerical attributes interpreted as strings, the values for attribute $a \in$QI are replaced with a string that includes their common prefix (substituting with a wildcard character the characters that differ). For example, values 10010, 10020, 10030 for attribute `ZIP` can be generalized to 100$**$, maintaining common prefix 100 and redacting the last two characters.
- *Set definition:* applicable to both categorical and numerical attributes, the values for attribute $a \in$QI are replaced with the set of values including all of them. For example, values *Italy*, *Italy*, and *France* for attribute `Country` can be generalized to the set $\{$*Italy*, *France*$\}$.
- *Interval definition:* applicable to numerical attributes defined on a totally ordered domain, the values for attribute $a \in$QI are replaced with a range of values including all of them. While the smallest range containing all the values to be generalized (i.e., the one delimited by the minimum and maximum value) is

the most natural choice, also larger (possibly predefined) intervals may be adopted. Note that this generalization is different from set definition: while the set definition explicitly maintains all the (original) values generalized in a set, interval definition maintains only the extremes of the range. To illustrate, consider the dataset in Figure 2(a) and suppose, for simplicity, it is a fragment retrieved by a worker for anonymization. Its anonymized version in Figure 2(c) is obtained generalizing attribute `Age` by grouping its values in intervals, considering the partitions in Figure 2(b). For example, values 42, 50, and 43 (fourth, fifth, and sixth tuples, corresponding to the partition on the right-hand-side of Figure 2(b)) are generalized to [42,50].



Age	Country	TopSpeed
25	Italy	132
...	...	...
38	USA	140

Age	Country	TopSpeed
25	Italy	132
...	...	...
30	France	128

Age	Country	TopSpeed
[25 - 30]	Europe	132
...	...	...
[25 - 30]	Europe	128

Fig. 8: Spark-based distributed anonymization system

6 Wrap up and information loss assessment

The wrap-up phase of our approach is aimed at collecting anonymized fragments, and at assessing information loss. To this purpose, each worker stores the anonymized fragment $\hat{F}$ assigned to it in the storage platform, and computes the information loss implied by the anonymization of F. The information loss characterizing the anonymized fragments are collected and combined by the Coordinator to assess the information loss of the entire dataset. In the following, we describe the information loss metrics we adopt. For the sake of readability, we refer to a dataset D and its anonymized version $\hat{D}$, with the note that each worker operates on the fragment F (and its anonymization $\hat{F}$) assigned to it.

- *Discernibility Penalty* (DP [7], [8]) assigns a *penalty* to each tuple in D based on the size of the equivalence class E (the larger an equivalence class, the larger the penalty) to which the tuple belongs (i.e., number of tuples generalized to the same values). Formally, the Discernibility Penalty of an anonymized dataset $\hat{D}$ is computed as follows:

$$\mathsf{DP}(\hat{D}) = \sum_{E \in \hat{D}} |E|^2 \tag{2}$$

- *Normalized Certainty Penalty* (NCP [9]) assigns penalties based on the amount of generalization (more generalization resulting in higher penalties) applied to the values of the quasi-identifying attributes. NCP is applicable to numerical attributes generalized in intervals, and to categorical attributes for which a generalization hierarchy exists. Given a tuple $t \in D$, the normalized certainty penalty $\mathsf{NCP}_a(\hat{t})$ of its generalization $\hat{t} \in \hat{D}$ for attribute $a \in \mathsf{QI}$ is computed as follows:

$$\mathsf{NCP}_a(\hat{t}) = \begin{cases} \frac{v_{max}-v_{min}}{Range(a)} & \text{if } a \text{ is numerical} \\ \frac{|Ind(\hat{v})|}{|Dom(a)|} & \text{if } a \text{ is categorical} \end{cases} \tag{3}$$

where $\hat{t}[a]$=$[v_{max}, v_{min}]$ and $Range(a)$ is the range of the values assumed by a, if a is a numerical attribute generalized in intervals; $\hat{t}[a]$=$\hat{v}$ and $Ind(\hat{v})$ is the set of values in $Dom(a)$ that could be generalized to $\hat{v}$ (i.e., the number of leaves of the generalization hierarchy $\mathcal{H}(a)$ for a that are descendants of $\hat{v}$), if a is a categorical attribute with generalization hierarchy. Given an anonymized dataset $\hat{D}$ with quasi-identifier QI, the Normalized Certainty Penalty of $\hat{D}$ is computed summing the Normalized Certainty Penalties of the attributes in QI for all the tuples in $\hat{D}$ as follow:

$$\mathsf{NCP}(\hat{D}) = \sum_{\hat{t} \in \hat{D}} \sum_{a \in \mathsf{QI}} \mathsf{NCP}_a(\hat{t}) \tag{4}$$

Given the information loss measures $\mathsf{DP}(\hat{F}_1), \ldots, \mathsf{DP}(\hat{F}_{|W|})$ ($\mathsf{NCP}(\hat{F}_1), \ldots, \mathsf{NCP}(\hat{F}_{|W|})$, resp.) for the fragments, the Coordinator can compute the information loss for the whole dataset by simply summing them. We consider both DP and NCP metrics since they take a different approach in the assessment: DP is independent from the amount of generalization adopted and only considers the size of equivalence classes, while NCP precisely assesses the amount of generalization, regardless of the number of tuples in equivalence classes.

7 Implementation

In this section, we illustrate the architectural design and the deployment of our distributed anonymization approach. The implementation is available at https://github.com/mosaicrown/mondrian.

7.1 Architecture

Our implementation is based on an *Apache Spark* cluster, whose nodes perform the three phases (pre-processing, anonymization, and wrap-up) of our approach. Figure 8 illustrates the components and working of our implementation. The dataset $\mathcal{D}$ to be anonymized can be stored on any storage platform (centralized or distributed) reachable by Apache Spark with a URL. The Spark cluster includes a *Spark Cluster Manager*, which coordinates the cluster, and a set W of *Spark Workers*, which perform the tasks assigned to them by the Cluster Manager. We implemented our distributed anonymization application in Python to leverage Pandas framework [10], [11], which can be conveniently used for managing very large datasets.

In the Spark cluster architecture in Figure 8, the *Spark Driver* plays the role of our Coordinator: it is responsible for calling the *Spark Context* in the user-written code, implementing the partitioning process. (Note that the Spark Driver and the Spark Cluster Manager are not necessarily hosted on the same node of the cluster.) The Spark Driver is also responsible for translating the code to be executed in the cluster into jobs, which are further divided into smaller execution units, called *tasks* (implementing the distributed anonymization in our scenario), executed by the workers.

To anonymize a dataset $\mathcal{D}$, first the Spark Driver downloads from the storage platform a sample D of $\mathcal{D}$ that fits into its memory. It then locally executes the pre-processing phase. In particular, the Spark Driver locally partitions the downloaded sample D running procedure **Q_Partition** in Figures 4 (if quantile-based approach is adopted), or procedure **M_Partition** in Figure 5 (if multi-dimensional approach is adopted), keeping track of the conditions describing the computed fragments. The Spark Driver then defines a set of $|W|$ Spark Tasks. Each task corresponds to the anonymization of a fragment F_i and includes the conditions defining F_i. Once the Spark Driver has terminated the pre-processing phase, the Spark Cluster Manager selects the workers that will be involved in the distributed anonymization process. The Spark Driver then sends to each identified worker w_i its anonymization task, enabling w_i to download from the storage platform the tuples in the fragment F_i assigned to it. Each Spark Worker w then retrieves from the storage platform the fragment of $\mathcal{D}$ satisfying the conditions included in the task assigned to it. The Spark Worker anonymizes the fragment with our distributed Mondrian (Figure 7, Section 5), computes the amount of information loss caused by anonymization (Equations 2 and 4, Section 5), and stores the results in the storage platform using the services of the Spark Driver. The Spark Driver combines the information loss computed by the Spark Workers to obtain the information loss of the overall dataset.

7.2 Deployment

Aiming at creating a solution that can be easily deployed in a cloud infrastructure (e.g., AWS or Google Cloud), we opted for a multi-container application, leveraging Docker containers, for the deployment of our approach. We deployed the Spark architecture in Figure 8 with: *i)* a Docker container for the Spark Driver; *ii)* a Docker container for the Spark Cluster Manager; *iii)* a variable number of Docker containers for the Spark Workers; and *iv)* a Docker container to expose a *Spark History Server*, for additional information about the task scheduling and assignment performed by Spark.

In our implementation, Docker containers are spawned with Docker Compose. To manage the distribution of the containers to the nodes (machines) of the Spark cluster, different orchestrator tools (e.g., Kubernetes) can be adopted. We leveraged Docker Swarm due to its simplicity. Figure 9 illustrates an example of the Docker Swarm distribution of Docker containers to the nodes of a Spark cluster. In the figure, white solid boxes represent the nodes in the Spark cluster, while blue (gray, in b/w printouts) boxes represent Docker containers. Note that each node in the Spark cluster can spawn more than one Docker container. One of the nodes in the cluster acts as *Docker Swarm Manager*, while the other nodes act as *Docker Swarm Workers*. The Docker Swarm Manager coordinates and is in charge of distributing the workload on the Docker Swarm Workers, which in turn are used to spawn the Docker containers modeling Spark Manager, Spark Driver, and Spark Workers. One of the Docker Swarm Workers is then dedicated to spawn one container for the Spark Cluster Manager and one container for the Spark Driver. The other Docker Swarm Workers spawn the containers modeling Spark Workers.

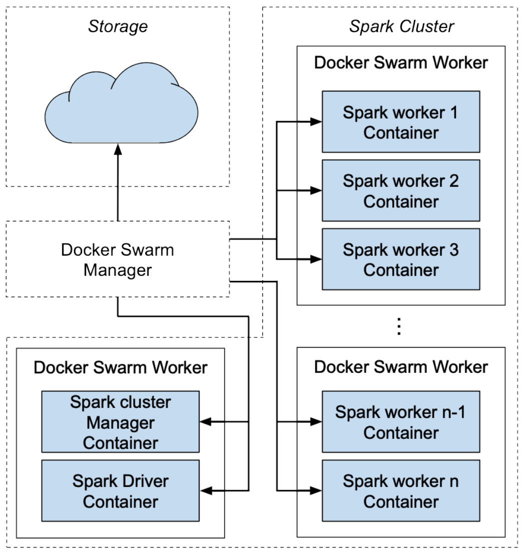


Fig. 9: Container deployment in a cloud environment

8 Experimental results

We performed a set of experiments to evaluate the scalability and applicability of our distributed Mondrian anonymization approach, compared to the traditional centralized Mondrian algorithm, considered as the baseline.

8.1 Experimental settings

In the following, we describe the settings and dataset used in our experimental evaluation.

Server specifications and cloud deployment. Since our solution does not require any specific cloud environment, to obtain reproducible results, in our experiments we simulated a cloud environment leveraging Docker Compose. We run our experiments on a machine equipped with an AMD Ryzen 3900X CPU (12 physical cores, 24 logical cores), 64 GB RAM and 2 TB SSD, running Ubuntu 20.04 LTS, Apache Spark 3.0.1, Hadoop 3.2.1, and Pandas 1.1.3. Each worker is equipped with 2GB of RAM and 1 CPU core. Centralized Mondrian relies on 1 CPU core, with no limitation on the use of the RAM. To prove the applicability and scalability of our solution in a real-world distributed environment, we deployed it on Amazon Elastic Compute Cloud (*t2.medium* instances equipped with 2 cores, 4GB of RAM, and 8GB of gp3 SSD, running Ubuntu 20.04 LTS, and located in the *us-east-1* region). The results obtained in this real-world cloud environment confirm the ones obtained in our simulated environment illustrated in Section 8.2.

Storage platform. We used *Hadoop Distributed File System (HDFS)* as the distributed storage platform for storing the dataset to be anonymized. We deployed the HDFS cluster leveraging Docker containers, with one container for the Hadoop Namenode (responsible for the cluster management); and multiple containers for the Hadoop Datanodes (responsible for storing data and servicing read and write requests).

Dataset. We considered the Poker Hand dataset [12] and ACS PUMS USA 2019 dataset [13]. The choice has been dictated by the need to consider very large datasets, to test the scalability of our approach. We refer the reader to [1] for performance and information obtained by our distributed Mondrian on the well known (but smaller) ACS PUMS USA 2018 dataset [13].

The Poker Hand dataset is composed of 1,000,000 tuples. Each tuple represents the cards in a hand of Poker. Each card is described through 2 attributes: the seed (an integer value in the range $\{1,\dots,4\}$) and the rank (an integer value in the range $\{1,\dots,13\}$). We considered these attributes as the quasi-identifier. We considered an additional attribute identifying the entire hand (an integer value in the range $\{0,\dots,9\}$) as the sensitive attribute.

For the ACS PUMS USA 2019 dataset, we extracted a sample of 1,500,000 tuples. Each tuple of the dataset represents an individual respondent with attributes `ST`, `OCCP`, `AGEP`, and `WAGP`, representing respectively the respondent's US State of residence, occupational status (expressed with a numeric code), age, and annual income. We considered `ST`, `OCCP`, `AGEP` as the quasi-identifier, and `WAGP` as the sensitive attribute. While `OCCP`, `AGEP`, and `WAGP` are numeric attributes, `ST` is categorical. For attribute `ST`, we consider a generalization hierarchy $\mathcal{H}(\texttt{ST})$ defined according to the criteria adopted by the US Census Bureau [14], where States are at the leaf level of the hierarchy and are grouped in Divisions, which are in turn grouped in Regions. For example, States NJ, NY, and PA (leaf level) can be generalized to MiddleAtlantic (which is their parent in the hierarchy), which in turn can be generalized to Northeast, which in turn can be generalized to US (which is the root of the hierarchy).

8.2 Results

To assess the scalability of our approach, we analyzed the computation time of our distributed Mondrian varying the number of workers. Also, to assess the quality of the solution computed by our distributed Mondrian, we analyzed the information loss varying the size of the sample used for partitioning the dataset among workers. Both computation times and information loss values reported in this section have been obtained as the average over 5 runs. The values are compared with centralized Mondrian, considered as our *baseline*. In the following, we report and comment more in the details the results obtained over Poker Hand dataset, where the evolution exhibited by execution time and information loss is more visible. We note however that the results obtained with ACS PUMS USA 2019 dataset have a similar behavior.

Computation time. Figure 10 compares the computation times of our distributed Mondrian anonymization algorithm over Poker Hand dataset, using both quantile-based and multi-dimensional partitioning, with centralized Mondrian anonymization, varying the number of workers and parameters k and ℓ. In particular, we considered k varying in $\{5, 10, 20\}$, ℓ varying in $\{2, 3, 4\}$, and a number of workers between 2 and 12 workers (12 is the largest number of partitions that quantile-based partitioning can produce due to the domain of the attributes of the dataset). As expected, the execution time decreases when the number of workers grows, with savings with respect to centralized Mondrian between 28% and 98%, confirming the scalability of our distributed approach. (We note that similar results were observed on the ACS PUMS USA 2019 dataset, with savings between 45% and 98%.) As visible from the figures, the chosen pre-processing strategy does not significantly affect computation times. Quantile-based and multi-dimensional partitioning exhibit the same execution time when using two workers. Indeed, both approaches would perform the same partitioning. We also note that, when using quantile-based partitioning, each additional worker provides a saving in computation time. On the contrary, when using multi-dimensional partitioning, computation time saving can be enjoyed when the number of workers reaches a power of 2 (i.e., it is necessary to have 2^i workers for saving on computation time). For instance, there is a marginal saving when passing from 6 to 7 workers, but there is a considerable saving from 7 to 8 workers. This is due to the fact that, when the number of workers is not a power of 2, some workers are assigned twice the workload as the others (see Section 4.1) and therefore represent a bottleneck.

Information loss. Our distributed Mondrian might cause additional information loss compared to the centralized Mondrian, since each worker independently operates on its fragment without coordinating with other workers. We observe that the information loss caused by distribution can be impacted by: 1) the number of workers (and hence of fragments), and 2) the size of the sample used to partition the dataset. Figure 11 compares the average information loss (and its variance) obtained in 5 runs of the distributed (with 5 and 10 workers) Mondrian with the average information loss of the centralized Mondrian for computing a k-anonymous (with $k = 5$, $k = 10$, and $k = 20$) 2-diverse version of the Poker Hand dataset, assuming different sampling sizes (0.1%, 0.01%, 0.001%). While in the figure we report only the values obtained with $\ell = 2$ for Poker Hand dataset, we tested also $\ell = 3$ and $\ell = 4$, also considering ACS PUMS USA 2019, obtaining similar results. The results in Figure 11 show that, for all values of k, sampling has a very limited impact on information loss. The results also confirm that, for all values of k, also the number of workers has negligible impact on information loss. More precisely, multi-dimensional partitioning performs similarly for all tested numbers of workers, while quantile-based partitioning produces higher information loss when the number of workers grows. The values in Figure 11 for DP reveal that multi-dimensional partitioning produces equivalence classes similar to the ones produced by centralized Mondrian, while the quantile-based approach produces slightly

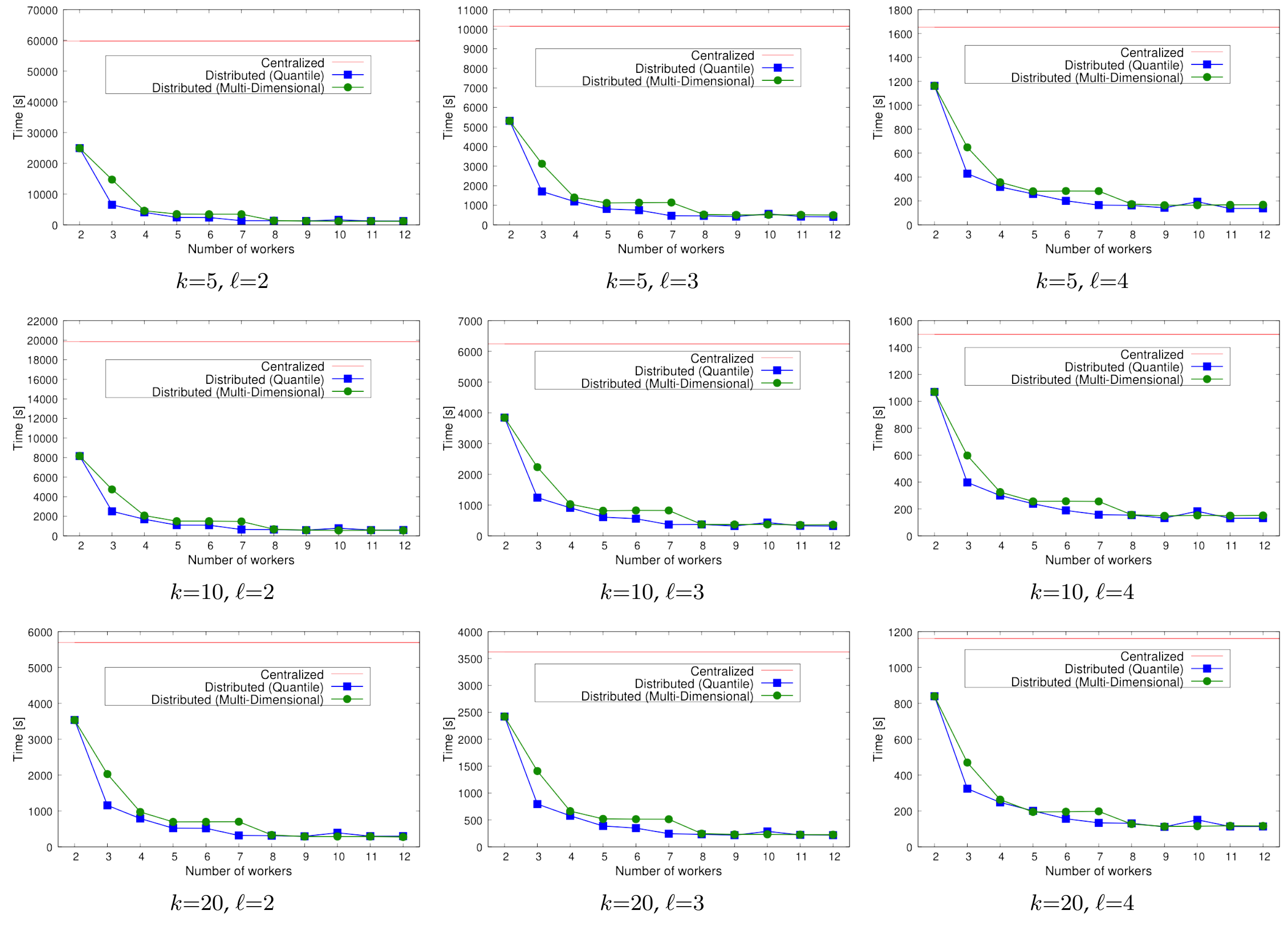


Fig. 10: Execution times of centralized Mondrian and distributed Mondrian varying the number of workers, k, and ℓ

smaller equivalence classes, especially when the sample used for partitioning is small. The results also show that, in some of the tested scenarios, the values for DP are higher in the centralized scenario. Even if the difference is negligible, it reveals that, when using sampling for partitioning the dataset, (a subset of) the equivalence classes are smaller compared to the equivalence classes obtained without sampling. Smaller equivalence classes, however, do not imply less generalization, as testified by the values of NCP. Indeed, quantile-based partitioning, multi-dimensional partitioning, and the centralized algorithm present similar (small) values for NCP.

The experiments confirm that our distributed Mondrian provides high scalability, while causing limited impact on information loss. When the number of workers is a power of 2, multi-dimensional and quantile-based partitioning exhibit similar computation time. However, when the number of workers is not a power of 2, quantile-based partitioning provides better performance. Both approaches have limited impact on information loss, with multi-dimensional partitioning having smaller values for NCP and higher values for DP than quantile-based partitioning (and vice versa).

9 Related work

The problem of protecting privacy in data publishing has been widely studied (e.g., [3], [15], [16], [17], [18]). The solutions proposed in the literature include both syntactic (e.g., k-anonymity [3] and ℓ-diversity [19]) and semantic techniques (e.g., differential privacy [15] and its variations [20]). Traditional algorithms aimed at enforcing k-anonymity and/or ℓ-diversity (e.g., [7], [21], [22]) operate in centralized scenarios. The problem of distributing and parallelizing anonymization has been recently studied, to the aim of protecting also large datasets (e.g., [23], [24]). The approach in [23] partitions the dataset and anonymizes the resulting fragments, leveraging MapReduce [25] paradigm to parallelize the centralized anonymization solution in [26]. The proposal in [23] takes a different approach in partitioning with respect to ours, since it aims at maintaining in each fragment the same value distribution as the whole dataset while we aim at maintaining in each fragment homogeneous values for the quasi-identifier, so to reduce the amount of generalization needed to enforce k-anonymity (and hence the information loss due to generalization). The distributed anonymization approach in [24] partitions data so that fragments contain records that are semantically similar (leveraging Locally Sensitive Hashing, semantic distance

		Information Loss (DP)		**Information Loss (NCP)**	
Sampling	**Partitioning**	**5 workers**	**10 workers**	**5 workers**	**10 workers**
0.1%	Quantile	7.14e06 ± 5.13e02	7.10e06 ± 8.31e03	1.83e06 ± 3.18e02	1.97e06 ± 3.99e02
	Multi-dimensional	7.22e06 ± 1.63e04	7.22e06 ± 9.25e03	1.80e06 ± 2.46e03	1.80e06 ± 2.15e03
0.01%	Quantile	7.14e06 ± 2.78e04	7.10e06 ± 9.35e03	1.83e06 ± 3.41e03	1.96e06 ± 1.01e04
	Multi-dimensional	7.17e06 ± 1.93e04	7.15e06 ± 1.36e04	1.79e06 ± 4.02e03	1.80e06 ± 7.44e03
0.001%	Quantile	7.14e06 ± 2.78e04	7.13e06 ± 1.27e04	1.83e06 ± 3.41e03	1.91e06 ± 7.92e02
	Multi-dimensional	7.20e06 ± 5.02e04	7.20e06 ± 5.02e04	1.80e06 ± 3.82e03	1.80e06 ± 3.82e03
Centralized		7.23e06		1.50e06	

(a) k=5, ℓ=2

		Information Loss (DP)		**Information Loss (NCP)**	
Sampling	**Partitioning**	**5 workers**	**10 workers**	**5 workers**	**10 workers**
0.1%	Quantile	1.42e07 ± 7.44e03	1.41e07 ± 9.94e03	2.20e06 ± 1.38e03	2.34e06 ± 1.14e03
	Multi-dimensional	1.42e07 ± 2.39e04	1.43e07 ± 1.47e04	2.17e06 ± 6.50e02	2.17e06 ± 8.62e02
0.01%	Quantile	1.42e07 ± 2.16e04	1.41e07 ± 7.92e03	2.20e06 ± 1.67e03	2.34e06 ± 1.10e04
	Multi-dimensional	1.42e07 ± 2.35e04	1.42e07 ± 1.71e04	2.17e06 ± 4.90e03	2.17e06 ± 4.29e03
0.001%	Quantile	1.42e07 ± 2.16e04	1.41e07 ± 2.95e04	2.20e06 ± 1.67e03	2.24e06 ± 8.00e04
	Multi-dimensional	1.43e07 ± 2.36e04	1.47e07 ± 2.36e04	2.17e06 ± 4.67e03	2.17e06 ± 4.51e03
Centralized		1.43e07		1.82e06	

(b) k=10, ℓ=2

		Information Loss (DP)		**Information Loss (NCP)**	
Sampling	**Partitioning**	**5 workers**	**10 workers**	**5 workers**	**10 workers**
0.1%	Quantile	2.88e07 ± 2.82e04	2.86e07 ± 2.06e04	2.50e06 ± 3.30e01	2.66e06 ± 1.55e03
	Multi-dimensional	2.88e07 ± 4.44e04	2.88e07 ± 5.77e04	2.47e06 ± 1.56e03	2.46e06 ± 4.32e03
0.01%	Quantile	2.88e07 ± 6.73e04	2.85e07 ± 4.21e04	2.50e06 ± 1.32e03	2.65e06 ± 1.76e04
	Multi-dimensional	2.88e07 ± 5.03e04	2.88e07 ± 1.86e04	2.47e06 ± 5.75e03	2.46e06 ± 4.00e03
0.001%	Quantile	2.88e07 ± 6.73e04	2.86e07 ± 2.86e04	2.50e06 ± 1.32e03	2.63e06 ± 3.43e03
	Multi-dimensional	2.88e07 ± 7.23e04	2.88e07 ± 6.83e04	2.47e06 ± 8.32e03	2.46e06 ± 6.17e03
Centralized		2.88e07		2.07e06	

(c) k=20, ℓ=2

Fig. 11: DP and NCP information loss varying the number of workers and k

measure, and k-member clustering), but it does not leverage Mondrian for computing fragments.

Different distributed anonymization approaches rely, similarly to our proposal, on distributed architectures for parallelization (e.g., [6], [27], [28], [29], [30], [31], [32]). The approach in [6] parallelizes Mondrian considering Apache Spark, but relies on data exchange among workers to coordinate anonymization of different portions of the original dataset distributed to workers. Our approach instead aims at limiting data exchange among workers. The solution in [27] differs from ours since it uses hierarchical clustering and k-means to provide ℓ-diversity instead of performing partitioning according to Mondrian strategy. The approach in [28] considers Apache Spark for parallelizing different anonymization approaches, but does not discuss the Spark-based adaptation of Mondrian. The solution in [29] randomly assigns tuples to workers, while our solution specifically studies a strategy for distributing tuples to workers to minimize information loss. The first approach aimed at parallelizing Mondrian algorithm has been proposed in [30] and is based on MapReduce paradigm. This solution differs from ours since it heavily relies on data exchanges among workers, while we aim at minimizing the need for workers to communicate for anonymization purposes so to reduce the delays and costs inevitably entailed by exchanging data over a network. A more recent approach relying on MapReduce for parallelizing Mondrian algorithm has been proposed in [31]. Besides the use of MapReduce in contrast to Apache Spark, this solution differs from ours in the strategy adopted for splitting the dataset among workers. The approach in [31] operates on the whole dataset (and not on a sample of the same) and adopts a distributed algorithm for partitioning, using a tree structure shared among the workers in the cluster, while we specifically aim at limiting inter-worker exchanges and coordination to improve performance. Also, the proposal in [31] does not use quantiles for partitioning. The proposal in [32], which enforces Mondrian using Spark, is complementary to ours as it focuses on improving performances by optimizing data structures used by Spark. We instead propose a novel distributed enforcement approach for Mondrian, which ensures scalability with limited information loss through a careful data partitioning design.

The idea of distributing the execution of Mondrian for anonymizing a large dataset based on partitioning a sample of the dataset, while limiting data exchange among workers, has been proposed by us in [1], [2]. In this paper, we considerably extend (both theoretically and experimentally) our prior work in [1], [2] with the definition of, and support for, novel partitioning strategies, generalization approaches, and metrics for selecting the most suitable attributes for data partitioning and for computing information loss.

A related line of work has investigated the anonymization of distributed data and/or multiple datasets (e.g., [33], [34], [35], [36]). While related, these solutions address a different problem, characterized by multiple sources of data (e.g., [33], [34]), possibly with multiple privacy requirements (e.g., [35]) and with different owners, each with visibility on its own portion of data (e.g., [36]).

Other solutions based on data fragmentation for anonymization use vertical fragmentation of the private table. Fragmentation operates in such a way to enforce ℓ-diversity (e.g., [37]) or to protect sensitive associations among attributes in the relation schema (e.g., [38], [39]).

10 Conclusions

We presented a scalable approach for distributed anonymization of very large datasets. Our approach partitions a dataset to be anonymized in fragments which are then distributed to multiple workers operating in parallel and independently. We proposed different partitioning strategies operating on a sample of the dataset, and a distributed version of Mondrian anonymization algorithm aimed at limiting information exchange among workers. The experimental results confirm that our approach is scalable and does not affect the quality of the computed solution in terms of information loss caused by anonymization.

Acknowledgements

This work was supported in part by the EC under projects MOSAICrOWN, MARSAL and GLACIATION, by the Italian Ministry of Research within the PRIN program under project HOPE, and by JPMorgan Chase & Co under project “k-anonymity for AR/VR and IoT/5G”.

References

[1] S. De Capitani di Vimercati, D. Facchinetti, S. Foresti, G. Oldani, S. Paraboschi, M. Rossi, and P. Samarati, “Scalable distributed data anonymization,” in *Proc. of IEEE PerCom 2021*, Kassel, Germany, March 2021.

[2] ——, “Artifact: Scalable distributed data anonymization,” in *Proc. of IEEE PerCom 2021*, Kassel, Germany, March 2021.

[3] P. Samarati, “Protecting respondents’ identities in microdata release,” *IEEE TKDE*, vol. 13, no. 6, pp. 1010–1027, November/December 2001.

[4] V. Ciriani, S. De Capitani di Vimercati, S. Foresti, and P. Samarati, “k-Anonymity,” in *Secure Data Management in Decentralized Systems*, T. Yu and S. Jajodia, Eds. Springer-Verlag, 2007.

[5] A. Machanavajjhala, J. Gehrke, and D. Kifer, “ℓ-diversity: Privacy beyond k-anonymity,” in *Proc. of ICDE 2006*, Atlanta, GA, USA, April 2006.

[6] F. Ashkouti, K. Khamforoosh, and A. Sheikhahmadi, “DI-Mondrian: Distributed improved Mondrian for satisfaction of the ℓ-diversity privacy model using Apache Spark,” *Information Sciences*, vol. 546, pp. 1–24, 2021.

[7] K. LeFevre, D. DeWitt, and R. Ramakrishnan, “Mondrian multidimensional k-anonymity,” in *Proc. of ICDE 2006*, Atlanta, GA, USA, April 2006.

[8] R. J. Bayardo and R. Agrawal, “Data privacy through optimal k-anonymization,” in *Proc. of ICDE 2005*, Tokoyo, Japan, April 2005.

[9] J. Xu, W. Wang, J. Pei, X. Wang, B. Shi, and A.-C. Fu, “Utility-based anonymization for privacy preservation with less information loss,” *ACM SIGKDD Explorations Newsletter*, vol. 8, no. 2, pp. 21–30, 2006.

[10] W. McKinney, “Data structures for statistical computing in Python,” in *Proc. of SciPy 2010*, Austin, TX, USA, July 2010.

[11] J. Reback *et al.*, “pandas-dev/pandas: Pandas,” February 2020, https://doi.org/10.5281/zenodo.3509134.

[12] R. Cattral and F. Oppacher, “Poker Hand Data Set, UCI machine learning repository,” 2007, https://archive.ics.uci.edu/ml/datasets/Poker+Hand.

[13] S. Ruggles, S. Flood, R. Goeken, J. Grover, E. Meyer, J. Pacas, and M. Sobek, “IPUMS USA: Version 10.0 [dataset],” Minneapolis, MN: IPUMS, 2020, https://doi.org/10.18128/D010.V10.0.

[14] “Census regions and divisions of the united states,” 2020. [Online]. Available: https://www2.census.gov/geo/pdfs/maps-data/maps/reference/us_regdiv.pdf

[15] C. Dwork, “Differential privacy,” in *Proc. of ICALP 2006*, Venice, Italy, July 2006.

[16] J. Vaidya and C. Clifton, “Privacy preserving association rule mining in vertically partitioned data,” in *Proc. of ACM KDD 2002*, Edmonton, Alberta, Canada, July 2002.

[17] J. Domingo-Ferrer and V. Torra, “A critique of k-anonymity and some of its enhancements,” in *Proc. of ARES 2008*, Barcelona, Spain, March 2008.

[18] ——, “Ordinal, continuous and heterogeneous k-anonymity through microaggregation,” *Data Mining and Knowledge Discovery*, vol. 11, no. 2, pp. 195–212, 2005.

[19] A. Machanavajjhala, D. Kifer, J. Gehrke, and M. Venkitasubramaniam, “ℓ-diversity: Privacy beyond k-anonymity,” *ACM TKDD*, vol. 1, no. 1, pp. 3:1–3:52, 2007.

[20] C. Dwork and A. Roth, “The algorithmic foundations of differential privacy,” *Foundations and Trends in Theoretical Computer Science*, vol. 9, no. 3-4, pp. 211–407, 2014.

[21] K. LeFevre, D. DeWitt, and R. Ramakrishnan, “Incognito: Efficient full-domain k-anonymity,” in *Proc. of SIGMOD 2005*, Baltimore, MA, USA, June 2005.

[22] B. Hore, R. C. Jammalamadaka, and S. Mehrotra, “Flexible anonymization for privacy preserving data publishing: A systematic search based approach,” in *Proc. of SIAM SDM 2007*, Minneapolis, MN, USA, April 2007.

[23] X. Zhang, L. T. Yang, C. Liu, and J. Chen, “A scalable two-phase top-down specialization approach for data anonymization using MapReduce on cloud,” *IEEE TPDS*, vol. 25, no. 2, pp. 363–373, 2013.

[24] X. Zhang, C. Leckie, W. Dou, J. Chen, R. Kotagiri, and Z. Salcic, “Scalable local-recoding anonymization using locality sensitive hashing for big data privacy preservation,” in *Proc. of CIKM 2016*, Indianapolis, IN, USA, October 2016.

[25] J. Dean and S. Ghemawat, “MapReduce: Simplified data processing on large clusters,” *CACM*, vol. 51, no. 1, pp. 107–113, 2008.

[26] B. Fung, K. Wang, and P. Yu, “Top-down specialization for information and privacy preservation,” in *Proc. of ICDE 2005*, March, Tokyo, Japan 2005.

[27] F. Ashkouti, K. Khamforoosh, A. Sheikhahmadi, and H. Khamfroush, “DHkmeans-ℓ-diversity: distributed hierarchical k-means for satisfaction of the ℓ-diversity privacy model using Apache Spark,” *The Journal of Supercomputing*, vol. 78, no. 6, pp. 2616–2650, 2021.

[28] S. Antonatos, S. Braghin, N. Holohan, Y. Gkoufas, and P. Mac Aonghusa, “Prima: an end-to-end framework for privacy at scale,” in *Proc. of ICDE 2018*, Paris, France, April 2018.

[29] U. Sopaoglu and O. Abul, “A top-down k-anonymization implementation for Apache Spark,” in *Proc. of IEEE Big Data 2017*, Boston, MA, USA, December 2017.

[30] A. Chakravorty, T. W. Wlodarczyk, and C. Rong, “A scalable k-anonymization solution for preserving privacy in an aging-in-place welfare intercloud,” in *Proc. of IC2E 2014*, Boston, MA, USA, March 2014.

[31] X. Zhang, L. Qi, W. Dou, Q. He, C. Leckie, R. Kotagiri, and Z. Salcic, “MRMondrian: Scalable multidimensional anonymisation for big data privacy preservation,” *IEEE TBD*, vol. 8, no. 1, pp. 125–139, 2022.

[32] S. U. Bazai, J. Jang-Jaccard, and H. Alavizadeh, "A novel hybrid approach for multi-dimensional data anonymization for Apache Spark," *ACM TOPS*, vol. 25, no. 1, pp. 5:1–5:25, 2021.
[33] T. Tassa and E. Gudes, "Secure distributed computation of anonymized views of shared databases," *ACM TODS*, vol. 37, no. 2, pp. 1–43, 2012.
[34] F. Kohlmayer, F. Prasser, C. Eckert, and K. A. Kuhn, "A flexible approach to distributed data anonymization," *Journal of Biomedical Informatics*, vol. 50, pp. 62–76, 2014.
[35] X. Ding, Q. Yu, J. Li, J. Liu, and H. Jin, "Distributed anonymization for multiple data providers in a cloud system," in *Proc. of DASFAA 2013*, Wu Han, China, April 2013.
[36] P. Jurczyk and L. Xiong, "Distributed anonymization: Achieving privacy for both data subjects and data providers," in *Proc. of DBSec 2009*, Montreal, Canada, July 2009.
[37] X. Xiao and Y. Tao, "Anatomy: Simple and effective privacy preservation," in *Proc of VLDB 2006*, Seoul, South Korea, September 2006.
[38] S. De Capitani di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, and P. Samarati, "Fragments and loose associations: Respecting privacy in data publishing," *PVLDB*, vol. 3, no. 1, pp. 1370–1381, September 2010.
[39] S. De Capitani di Vimercati, S. Foresti, S. Jajodia, G. Livraga, S. Paraboschi, and P. Samarati, "Loose associations to increase utility in data publishing," *JCS*, vol. 23, no. 1, pp. 59–88, 2015.

Sabrina De Capitani di Vimercati is a professor at the Università degli Studi di Milano, Italy. Her research interests are in data security and privacy. She has published more than 210 papers in journals, conference proceedings, and books. She has been a visiting researcher at SRI International, CA, USA, and George Mason University, VA, USA.
https://decapitani.di.unimi.it

Dario Facchinetti is a post-doctoral researcher at the Università degli Studi di Bergamo, Italy. His work ranges from the integration of security features in mobile, database and cloud systems, to policy and privacy management. He is interested in access control and sandboxing techniques.

Sara Foresti is a professor at the Università degli Studi di Milano, Italy. Her research interests are in data security and privacy. She has published more than 100 papers in journals, conference proceedings, and books. She has been a visiting researcher at George Mason University, VA, USA. She chairs the IFIP WG 11.3 on Data and Application Security and Privacy.
https://foresti.di.unimi.it

Giovanni Livraga is an associate professor at the Università degli Studi di Milano, Italy. His research interests are in the area of data privacy and security in emerging scenarios. His PhD thesis received the ERCIM STM WG 2015 award. He has been a visiting researcher at SAP Labs, France and George Mason University, VA, USA.
https://livraga.di.unimi.it

Gianluca Oldani is currently pursuing the Ph.D. degree with the Università degli Studi di Bergamo, Italy. His research interests include web security, distributed technologies, and data privacy.

Stefano Paraboschi is a professor at the Università degli Studi di Bergamo, Italy. His research focuses on information security and privacy, Web technology for data intensive applications, XML, information systems, and database technology. He has been a visiting researcher at Stanford University and IBM Almaden, CA, USA, and George Mason University, VA, USA.
https://cs.unibg.it/parabosc

Matthew Rossi is currently pursuing the Ph.D. degree with the Università degli Studi di Bergamo, Italy. From 2019 to 2020, he was a Research Assistant with the Department of Information Engineering, Università degli Studi di Bergamo. His research interest includes the integration of security features in mobile systems, policy management and privacy of outsourced data.

Pierangela Samarati is a professor at the Università degli Studi di Milano, Italy. Her main research interests are in data protection, security, and privacy. She has published more than 280 papers in journals, conference proceedings, and books. She has been a visiting researcher at Stanford University, CA, USA, SRI International, CA, USA, and George Mason University, VA, USA. She is a Fellow of ACM, IEEE, and IFIP.
https://samarati.di.unimi.it