TBA

Marco Anisetti *Senior Member, IEEE,*, Claudio A. Ardagna *Senior Member, IEEE,* Alessandro Balestrucci, Chiara Braghin, Ernesto Damiani *Senior Member, IEEE,*, Antongiacomo Polimeno

Abstract—The conflict between the need of protecting and sharing data is hampering the spread of big data applications. Proper security and privacy assurance is required to protect data owners, while proper data access and sharing are fundamental to implement smart big data solutions. In this context, access control systems assume a central role for balancing the need of data protection and sharing. However, given the software and technological complexity of big data ecosystems, existing solutions are not suitable because they are neither general nor scalable, and do not support a dynamic and collaborative environment. In this paper, we propose an access control system that enforces access to data in a distributed, multi-party big data environment. It is based on data annotations and secure data transformations performed at ingestion time. We show the feasibility of our approach with a case study in a smart city domain using an Apache-based big data engine.

Index Terms—Access Control, Big Data, Data Transformation, Data Ingestion

1 Introduction

TBW

2 Motivations

2.1 Background

2.2 Privacy

2.3 Accuracy

3 System Model

4 Policy

4.1 Annotations

4.2 Transformation

4.3 Policy

5 Service Composition

5.1 Template

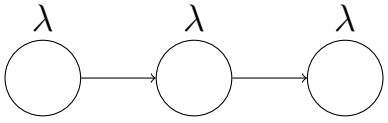

Fig. 1. Service composition template

- *M. Anisetti, C.A. Ardagna, E. Damiani, are with the Dipartimento di Informatica, Università degli Studi di Milano, Milano, Italy. E. Damiani is also with.*
E-mail: {firstname.lastname}@unimi.it

5.2 Instance

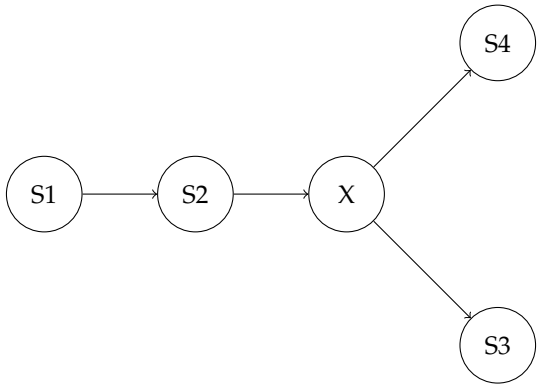

Fig. 2. Service composition instance

$$\forall S \in \mathrm{S}_C \exists \lambda(S) = \mathrm{S}_1$$

6 Coalition Building

Coalition building is a crucial process having direct impact on the quality of the analytics results. Figure **??** shows how data lineage is impacted by the processing lineage and in particular by i) the *coalition agreement* CA_C (i.e., the CA-driven transformations adopted for a give coalition) and by ii) the transformation produced by the different jobs (job-specific transformation) part of a given coalition C. Let us consider job $J_1^{o_1}$ of Figure **??** it receives as input the data $T_1(D_1)$ based on the dataset obtained by D_1 after the transformation T_1 which is associated to the data lineage by our AC model. It then produce a data that is the job-specific transformation on the input data (i.e., $T_1(D_1)$) generating D_2. We note that our Big Data Analytics pipeline models includes alternatives allowing different processing lineage (linear independent path in the Big data graph G) doing the same analytics but using different jobs (e.g., a lineage including k-means or a lineage using c-means). This will lead to different job-specific transformation on

the data for the same Big Data pipeline. In this paper, for the sake of simplicity we i) consider different coalitions for each processing lineage, ii) coalitions made of trustworthy organizations o_i providing candidate services for each job and iii) job-specific transformation not influenced by the organizations' behavior. In this scenario, since any coalition of a given processing lineage will produce the same job-specific data transformation, the analytics pipeline quality is impacted only by the *coalition agreement* CA_C or rather by the transformations T_i imposed by the given coalition C on the data lineage. In the following we first present metrics to evaluate data quality across the data lineage, and then a set of solutions to build coalitions for given Big Data pipeline ensuring a given data quality.

6.1 Metrics

Data quality is a largely studied topic for the database management research communities, and is in general focused on the quality of the data source rather then on the quality of the data outcomes or of the data while used in the processing pipeline. In [4] a survey on big data quality is proposed mentioning the well known categories of big data quality grouped by intrinsic, contextual representational and accessibility categories. It also presents an holistic quality management model where the importance of data quality during processing is just mentioned in terms of requirements for the pre-processing job (e.g., data enhancement due to cleaning jobs). In this paper we depart from this idea on data quality at pre processing time only measuring it at each step of the big data pipeline.

In the following we present a set of metrics to evaluate the quality of the data at each step of the big data pipeline. We

The proposed metrics can be classified into two categories, namely quantitative and statistical. Initially, these metrics are applied to the original dataset (X) without any transformations, and subsequently, they are applied to the transformed dataset (Y). The quantitative approach facilitates the calculation of the amount of data that has been lost during the transformation by enumerating the differences between X and Y. On the other hand, the statistical approach takes into consideration the changes in certain statistical properties before and after the transformation. These metrics can be applied either to the entire dataset or specific features. The features can be assigned either equal or varying weights, which enables the prioritization of important features that were lost during the transformation.

6.1.1 Jaqard coefficent

Let us consider two dataset X and Y of the same size. The Jaqard coefficent is defined as:

$$J(X,Y) = \frac{|X \cap Y|}{|X \cup Y|}$$

The use of Jaccard coefficient has several advantages when applied to datasets. Unlike other similarity measures, such as Euclidean distance, Jaccard coefficient is not affected by the magnitude of the values in the dataset. This property makes it suitable for datasets with categorical variables or nominal data, where the values do not have a meaningful numerical interpretation. The coefficient ranges from 0 to 1, where 0 indicates no similarity and 1 indicates complete similarity between the sets.

6.1.2 Jaqard coefficent with weights

Let us consider two dataset X and Y of the same size. The Jaqard coefficent is defined as:

$$J(X,Y) = \frac{\sum_{i=1}^{n} w_i(x_i \cap y_i)}{\sum_{i=1}^{n} w_i(x_i \cup y_i)}$$

Weighted Jaccard similarity is a variant of the Jaccard coefficient that incorporates weights to the elements in the sets being compared. It allows for the prioritization of certain features or elements in the datasets. This approach can be particularly useful when some elements in the dataset have more importance or relevance than others. By assigning weights to the elements, the weighted Jaccard similarity can account for this importance and provide a more accurate measure of similarity.

6.1.3 Kullback-Leibler divergence

Let us consider two dataset X and Y of the same size. The KL divergence is defined as:

$$KL(X,Y) = \sum_{i=1}^{n} x_i \log \frac{x_i}{y_i}$$

The KL divergence is a measure of the difference between two probability distributions and is useful for comparing the dissimilarity of two datasets.

6.1.4 Kullback-Leibler divergence with weights

Let us consider two dataset X and Y of the same size. The weighted KL divergence is defined as:

$$KL(X,Y) = \sum_{i=1}^{n} w_i(x_i \log \frac{x_i}{y_i})$$

The weighted KL divergence is a variant of the KL divergence that incorporates weights to the elements in the datasets being compared. It allows for the prioritization of certain features or elements in the datasets. This approach can be particularly useful when some elements in the dataset have more importance or relevance than others. By assigning weights to the elements, the weighted KL divergence can account for this importance and provide a more accurate measure of dissimilarity.

6.1.5 Jensen-Shannon Divergence

Let us consider two datasets X and Y of the same size. The Jensen-Shannon divergence (JSD) is a symmetrized version of the KL divergence and can be used to measure the dissimilarity between the two probability distributions.

The JSD between X and Y can be calculated as follows:

$$JSD(X,Y) = \frac{1}{2}\left(KL(X||M) + KL(Y||M)\right)$$

where M = 0.5 * (X + Y) is the average distribution.

The JSD incorporates both the KL divergence from X to M and from Y to M. It provides a balanced measure

of dissimilarity that is symmetric and accounts for the contribution from both datasets.

The JSD can be useful for comparing the dissimilarity of two datasets when you want a symmetric and normalized measure that considers the overall distribution of the data.

By utilizing the JSD, you can obtain a more comprehensive understanding of the dissimilarity between X and Y, taking into account the characteristics of both datasets.

6.1.6 Jensen-Shannon Divergence with Weights

Let us consider two datasets X and Y of the same size. The Jensen-Shannon divergence (JSD) can be extended to incorporate weights in the calculation, resulting in a weighted version of the divergence. This weighted Jensen-Shannon divergence accounts for the importance or relevance of specific features or elements in the datasets being compared.

The weighted Jensen-Shannon divergence between X and Y, denoted as JSDw(X, Y), is defined as:

$$JSDw(X,Y) = \frac{1}{2}\left(\sum_{i=1}^{n} w_i \left(x_i \log \frac{x_i}{m_i}\right) + \sum_{i=1}^{n} w_i \left(y_i \log \frac{y_i}{m_i}\right)\right)$$

where x_i and y_i are the elements of X and Y, respectively, w_i represents the weights assigned to each element, and

$$m_i = \frac{x_i + y_i}{2}$$

is the average of the corresponding elements.

By incorporating weights into the JSD calculation, the weighted Jensen-Shannon divergence provides a more accurate measure of dissimilarity between X and Y, considering the importance of individual elements based on the assigned weights. This approach is particularly useful when certain elements in the datasets have varying levels of significance, enabling a more tailored analysis of dissimilarity.

6.2 Metrica Composta tipo vikor

?

6.3 Metrica Composta vikor weighted (TOPSIS)

?

6.4 Problema

The metrics established will enable the quantification of data loss pre- and post-transformations. In the event of multiple service interactions, each with its respective transformation, efforts will be made to minimize the loss of information while upholding privacy and security standards. Due to the exponential increase in complexity as the number of services and transformations grow, identifying the optimal path is inherently an NP-hard problem. As such, we propose some heuristics to approximate the optimal path as closely as possible. To evaluate their efficacy, the heuristically generated paths will be compared against the optimal solution.

6.5 Prova di NP-hardness

6.6 Euristics

Greedy

The greedy algorithm is a heuristic that can be used to minimize the quantity of information lost by making locally optimal choices at each step in the hope of achieving a globally optimal solution. For instance, in our service selection problem, the greedy algorithm can be used to select services in order of their information loss, starting with the service with the lowest information loss. This strategy ensures that services with lower information loss are selected first, minimizing the overall quantity of information lost. Pseudocode for the greedy algorithm is presented in Algorithm 1.

```
function GreedyHeuristic(Set nodes):

  selectedNodes = empty
  while nodes is not empty:
    minMetricNode = None
    minMetricValue = infinity
    for node in nodes:
      currentMetric = calculateMetric(node)
      if currentMetric < minMetricValue:
        minMetricValue = currentMetric
        minMetricNode = node
        add minMetricNode to selectedNodes
        remove minMetricNode from nodes
  return selectedNodes
```

Listing 1. Greedy Heuristic Pseudocode

Sliding Window

The sliding window algorithm is a heuristic that can be used to minimize the quantity of information lost by considering a subset of the available services defined by a fixed or moving window. For example, in our service selection problem where the quantity of information lost needs to be minimized, the sliding window algorithm can be used to select services composition that have the lowest information loss within a fixed-size window. This strategy ensures that only services with low information loss are selected, minimizing the overall quantity of information lost. Pseudo-code for the sliding window algorithm is presented in Algorithm 2.

```
function SlidingWindowHeuristic(Seq sequence, int
    windowSize):

  selectedNodes = empty
  for i from 0 to length(sequence) - windowSize:
    minMetricNode = None
    minMetricValue = infinity
    for j from i to i + windowSize:
      currentMetric = calculateMetric(sequence[j])
      if currentMetric < minMetricValue:
        minMetricValue = currentMetric
        minMetricNode = sequence[j]
    add minMetricNode to selectedNodes
  return selectedNodes
```

Listing 2. Sliding Window Heuristic

The utilization of heuristics in service selection can be enhanced through the incorporation of techniques derived from other algorithms, such as Ant Colony Optimization or Tabu Search. By integrating these approaches, it becomes feasible to achieve a more effective and efficient selection of services, with a specific focus on eliminating paths that have previously been deemed unfavorable.

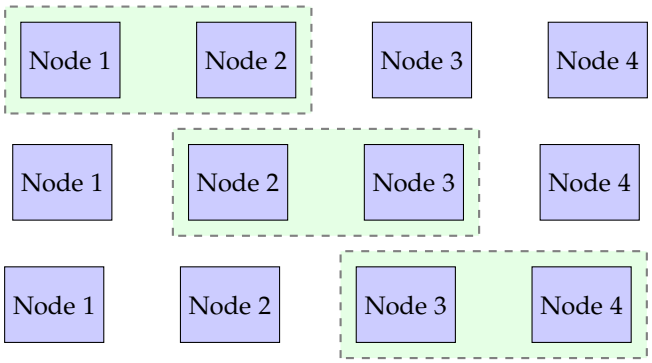

Fig. 3. Sliding Window Heuristic

title

7 Experiments

8 Related Work

- We believe that the closest approach to the one in this paper is the work of Hu et al. [2], introducing a generalised access control model for big data processing frameworks, which can be extended to the Hadoop environment. However, the paper discusses the issues only from a high-level architectural point of view, without discussing a tangible solution.
- [5] purpose-aware access control model, where purposes are data processing purpose and data operation purpose; the enforcement mechanism, still based on yes/no answer is based on an algorithm that checks if the operation on data to be performed matches to the purpose. The examples are given only for structured data and SQL queries. E se da una parte fa piu' di altri, dall'altra non ci sono attributi associati ai soggetti e agli oggetti, cosa che limita un pochino.
- [1] propose a solution specifically tailored to the Apache Hadoop stack, una semplice formalizzazione dell'AC in Hadoop. Non considerano la messa in sicurezza dell'ingestion time e non considerano la questione delle coalizioni. Considerano solo servizi all'interno di Hadoop ecosystem. Classica risposta yes/no.
- [3] questo e' solo su HBase

9 Conclusions

DISCUTERE CON MARCO POSSIBILI ANALOGIE SECURESCM: ad esempio i servizi non sono alternativi ma sono delle piccole coalizioni a loro volta.

(ri)Vedere anche Autocrmap per vedere analogie.

References

[1] Gupta, M., Patwa, F., Sandhu, R.: An Attribute-Based Access Control Model for Secure Big Data Processing in Hadoop Ecosystem. In: Proceedings of the Third ACM Workshop on Attribute-Based Access Control, ABAC'18, pp. 13–24. Association for Computing Machinery, New York, NY, USA (2018)

[2] Hu, V., Grance, T., Ferraiolo, D., Kuhn, D.: An Access Control Scheme for Big Data Processing. In: Proc. of 10th IEEE International Conference on Collaborative Computing: Networking, Applications and Worksharing, pp. 1–7. Miami, USA (2014)

[3] Huang, L., Zhu, Y., Wang, X., Khurshid, F.: An Attribute-Based Fine-Grained Access Control Mechanism for HBase. In: S. Hartmann, J. Küng, S. Chakravarthy, G. Anderst-Kotsis, A.M. Tjoa, I. Khalil (eds.) Database and Expert Systems Applications, pp. 44–59. Springer International Publishing (2019)

[4] Taleb, I., Serhani, M.A., Dssouli, R.: Big data quality: A survey. In: 2018 IEEE International Congress on Big Data (BigData Congress), pp. 166–173 (2018). DOI 10.1109/BigDataCongress.2018.00029

[5] Xue, T., Wen, Y., Luo, B., Zhang, B., Zheng, Y., Hu, e., Li, Y., Li, G., Meng, D.: GuardSpark++: Fine-Grained Purpose-Aware Access Control for Secure Data Sharing and Analysis in Spark. In: Proc. of ACM Annual Computer Security Applications Conference, ACSAC'20, pp. 582–596 (2020)