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 Requirements and System Model

Big data is highly dependent on cloud-edge computing, which makes extensive use of multi-tenancy. Multi-tenancy permits sharing one instance of infrastructures, platforms or applications by multiple tenants to optimize costs. This leads to common scenarios where a service provider offers subscription-based analytics capabilities in the cloud, or a single data lake is accessed by multiple customers. Thus, it is a common situation to have a big data pipeline where data and services belong to various organizations, posing a serious risk of potential privacy and security violation. In the following of this section, we present our system model (Section 2.1), the requirements driving our work (Section 2.2), and our reference scenario (Section 3).

2.1 System Model

Our system is a coalition of organizations that collaboratively execute a Big Data pipeline where *i)* organizations join without necessarily integrating their cloud-based or on-premises ICT infrastructures, *ii)* collaborative processes are carried out involving multi-party data collection and analytics, iii) the pipeline can be executed in a centralized or distributed deployment. It includes 4 different parties: *i)* the pipeline owners, executing Big Data analytics pipelines, *ii)* the organizations, providing the different services composing a big data pipeline, *iii)* the coalitions, as an orchestration of organizations providing all services at the basis of the Big Data analytics pipelines.

- *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

A big data pipeline can be defined as a composition of services, each one implementing a job, which can be parallelized to exploit the big data infrastructure capabilities. It defines executable processes that consist of a set of jobs combined using different structures to form a coherent system. Depending on the scope, jobs can be roughly classified as *i) ingestion jobs*, capturing and transforming data with the scope of saving it for further analysis, dealing with different data format (i.e., structured, unstructured, and semi-structured), *ii) analytics jobs*, executing specific analysis on data, which may include data preparation tasks (e.g., normalization, cleaning, selections) to make data suitable for specific analytics, *iii) visualization jobs*, presenting the output of an analytics to users.

We formally model a Big Data analytics pipeline as follows.

Definition 2.1 (Big Data Analytics Pipeline)**.** A Big Data Analytics pipeline $G(V,E)$ is a direct acyclic graph having a root $v_r \in V$, a vertex $v_i \in V_I \subseteq V$ for each job J_i invocation, two additional vertices $v_c, v_m \in V_{\otimes} \subset V$ for each alternative ($\otimes$) structure modeling the alternative execution (*choice*) of operations and the retrieval (*merge*) of the results, respectively, and two additional vertices $v_f, v_j \in V_{\oplus} \subset V$ for each parallel ($\oplus$) structure modeling the contemporary execution (*fork*) of operations and the integration (*join*) of their results, respectively.

We note that each vertex v_i model a job J_i provided by an organization o_i. We also note that an analytics pipeline can be deployed following a centralized or a decentralized approach as discussed in detail in Section 5.

Figure 1 shows an overview of our system model. In our model, organizations form coalitions, that is, collaborative networks of autonomous domains implementing a processing lineage of a given a Big Data analytics pipeline where resource sharing is achieved by the distribution of access permissions to coalition members based on negotiated resource-sharing agreements. A big Data pipeline can be constitute of a set of linear independent path (processing

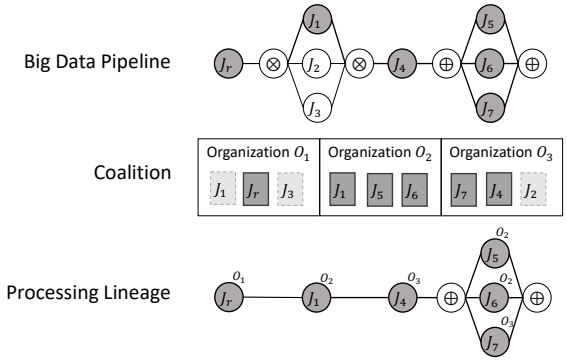

Fig. 1. Big Data Analytics pipeline graphs with a coalition of organization for a given processing lineage.

lineage) each of them implemented by coalition. In most cases, these coalitions are dynamic in nature, as changing conditions and trust relationships result in new missions and modifications to coalition membership. This scenario introduces a clear IT governance conflict: data should be compartmentalized to ensure strong protection, on one side, and shared to enable advanced analytics and key inter-organizational business processes, on the other side. This results in a set of strict requirements on existing access control systems that are discussed in the following section.

2.2 Requirements

The emerging need of balancing need to know/share and need to protect data in a coalition requires to rethink existing access control systems at the core of big data governance solutions. These solutions must consider the technical peculiarities of big data systems [1], [2], which point to scenarios where huge amount of data are diverse, come at high rates and must be proven to be trustworthy, as clarified by the 5V storyline [4]. In addition, they must take into account the increased governance complexity introduced by the collaborative environment. New requirements then emerge as follows.

[R1]: Authorization should be the primary focus. Authentication is assumed to be managed by a separate and integrated module to guarantee federations within big data ecosystem.

[R2]: Access control enforcement must protect data during their entire life cycle, along all processing phases, from ingestion to visualization, guaranteeing that data are properly protected and shared only to authorized users/organizations and for authorized operations.

[R3]: Along a big data pipeline, access to data must be evaluated before any operation on data takes place (guaranteeing a sort of *least privilege* at each processing step).

[R4]: Access control enforcement should support fine-grained access control, dealing with both structured and unstructured data. When structured data are considered, policies should refer to a (set of) single cell, a column, a tuple or an entire table of structured data. When unstructured data are considered, policies should specify the portion of the unstructured file whose access need to be regulated.

[R5]: Access control policies must support the specification of dynamic context-based access conditions, that is, access rights might depend on the run-time characteristics of the big data system.

[R6]: Access control policies must support the specification of dynamic access conditions based on the actual coalitions among organizations and their sharing agreement on both data and services.

[R7]: Dealing with a collaborative environment, access control enforcement should not use data ownership as the only attribute to define access rights, but rather consider the evolving state of resources and processing services within a specific big data context. CHIARA - Qui volevo esprimere il fatto che i servizi eseguiti lungo la pipe possono appartenere a domini/organizzazioni diverse, che possono anche non essere i proprietari della risorsa, ma non ci sono riuscita

[R8]: In collaborative big data scenarios, where data are shared among large coalitions of different organizations, sensitive data should be exposed only to the level required for specific business needs, and according to coalition's sharing agreement and coalition's member privileges.

[R9]: Access control should protect the privacy of sensitive data.

[R10]: In collaborative big data scenarios, traditional accept-deny enforcement is not always possible/wanted especially in latency-sensitive scenarios. Deny access to data should be modeled as a data preparation job, transforming data according to privileges modeled in access control policies.

[R11]: Access control enforcement should be highly efficient and scalable to accomplish the increasing cardinality of data and rate of requests.

[R12]: Access control enforcement should guarantee a proper balance between data share and data protection.

3 Reference Scenario

TBW

4 A Solution for Access Control in Big Data Scenarios

Our methodology is based on two key pillars: on one side, we define an *attribute-based access control model* that offers flexible fine grained authorization capabilities and exploits XACML notion of obligation to introduce preemptive data transformations that in a collaborative big data scenario are more suitable that denying data access. On the other side, differently from our previous work [3] where access control was done only at ingestion time, we monitor every data processing operation along the whole pipeline (see requirement **R3**).

4.1 Access Control Policy Model

Recently, attribute-based access control has gained a lot of attention thanks to its highly customized ability to express dynamic and fine grained rules. Instead of assigning capabilities (i.e., action/object pairs) directly to a subject or to its role, policies rules specify under which conditions access is granted or denied. Rules are specified in terms of attributes of the subject, attributes of the object and environment conditions, with the advantage of allowing policies to be created and managed without direct reference to potentially numerous subjects and objects, and subject

and objects to be provisioned without reference to a policy (requirement **R6**). This approach also minimizes the number of policy rules needed (requirement **R11**). Finally, by playing with attributes and environment conditions, we are able to express policies satisfying the requirements listed in Section 2.2 and the peculiarities of big data scenarios.

Our access control model then extends the policy structure of traditional attributed-based access control systems with two features:

- XACML optional obligations [7] specifying actions that must be performed before the policy is enforced and not directly associated with the controlled data become mandatory and are used to express data transformations that sanitize data resources to guarantee a minimum level of access to data.
- the evaluation of a policy rule always ends with an allow answer. Indeed, data protection is guaranteed by means of data transformations.

As in any attribute-based access control model, the key elements of our model are defined as follows:

- XACML optional obligations [7] specifying actions that must be performed before the policy is enforced and not directly associated with the controlled data become mandatory and are used to express data transformations that sanitize data resources to guarantee a minimum level of access to data (requirement **R8**).
- the evaluation of a policy rule always ends with an allow answer. Indeed, data protection is guaranteed by means of data transformations (requirement **R10**). Obviously, deny access to data can happen in extreme cases when data transformations delete all data resources[1].

As in any attribute-based access control model, the key elements of our model are defined as follows:

- A *subject* is a user or the service provider of a job that issues access requests to perform operations on objects. Subjects may have one or more attributes of the form (*aname*, *avalue*) pair, with *aname* the name of the attribute and *avalue* its value. In our case, there is always at least one attribute, specifying the organisation the subject belongs to. We use the notation *u.aname* to refer to the value of attribute named *aname*. For short, we use the notation $u@X$ to specify user u belonging to organization X, i.e., to express the value of *u.organization*. CHIARA - credo abbia senso specificare dei subject predefiniti tipo any, none e admin, anche dei service provider predefiniti come kafka, hbase, hive. Lo faccio qui o nell'esempio?
- An *object* is any data whose access is governed by the policy (requirement **R4**). It can be a file (e.g., a video, text file, image, etc.), a database, a table, a column, a raw, or a cell of a table. Like subjects, objects are assigned one or more attributes and have at least one, specifying the organization the object belongs to, expressed with the same notation $o@X$ for object o belonging to organization X.
- An *action* is any job that can be performed within a big data framework, ranging from classical atomic operations on a database (e.g, CRUD operations varying depending on the data model) to coarser operations such as an Apache Spark DAG, an Hadoop MapReduce, an analytics function call, or a pipe offered by a different service provider.
- The *environment* is defined by a set of dynamic attributes related to the specific context, such as time of the day, location, ip address, risk level, etc. CHIARA - mettere degli attributi piu' sensati, come normal e critical situation.

The access control policy is then defined as a set of policy rules expressing access conditions based on the key elements and their attributes. As mentioned earlier, the novelty of our model is in the exploitation of suitable data transformation functions that are applied to the target resource in order to sanitize it before access is given (requirement **R8**). As a consequence, when a subject makes an access request to a resource (i.e., an object), the policy rule matching the access request is evaluated, and the access is given after one or more data transformations are performed (requirement **R10**). Obviously, deny access to data can happen in extreme cases when data transformations delete all data resources[2].

Definition 4.1 (Policy Rule)**.** A *policy rule* is a predicate $policy\_rule_{obj}(subj, action, env, datatrans)$ defined as:

```
if (cond_expr == true)
then datatrans
```

where `cond_expr` is a propositional boolean expression built on composition of mathematical operators ($>, <, =, +, -, *, /$), logical operators ($\wedge, \vee, \neg$), set operators ($\in, \subset, \subseteq, \supset, \supseteq \cup, \cap$) and logical quantifiers ($\forall, \exists$) on the resource *obj* and on arguments *subj*, *action* and *env* and their attributes. `datatrans` are functions transforming data resources in different ways, mainly to protect against information leakage. CHIARA - da sistemare rendendo piu' leggibile, da decidere se va bene che i permessi siano dati come ACL

Transformation functions range from pruning and reshaping to encrypting/decrypting or anonymizing the full resource or part of it. Thus, data protection is obtained by removing or obfuscating sensitive attributes rather than denying access to full data. In section 4.3, we define a taxonomy of data transformations based on the security property that has to be guaranteed. The transformation function can be applied to the full resource or to part of it, either a single attribute or a set of attributes. For example, in case of structured data it can be applied to a single cell, to (one or more) rows or columns, either to the entire table. The target of the transformation function is given as parameter to the function.

In attribute-based access control model, policies are limited only by the computational language and the richness of the available attributes. In our case, we exploit the *environ-*

1. In case of a data transformation function returning an empty dataset, the pipeline is not broken.

2. In case of a data transformation function returning an empty dataset, the pipeline is not broken.

ment field of our model to contain the dynamic information on actual coalitions to express coalitions.

Definition 4.2 (Coalition)**.** A coalition C is a set of organizations $o \in O$, with O the set of the organizations associated to the specific scenario.

The *environment* field of our model then contains both O and the actual coalitions. Given an organization o o, function $Coalition(O)$ returns the set of coalitions an organization $o \in O$ is part of, i.e, $Coalition(o) = \{C | o \in C\}$. In a coalition resource sharing is achieved by the distribution of access permissions to coalition members. In order to simplify and speed up policy writing, it is common to define a default set of permissions rules to be applied to all coalition members. This set of rules are part of a coalition agreement.

Definition 4.3 (Coalition Agreement)**.** A *coalition agreement* CA_C is a set of policy rules associated to a coalition C that are added to the security policy of each organization.

Obviously, the rules of a coalition agreement depend on the specific nature of a coalition. For example, given a coalition C between organisations that do not handle sensitive data, a default rule in the coalition agreement could be:

$$policy\_rule_{obj}(subj, \texttt{any}, env, -) \equiv obj.organization \in C \wedge subj.organization \in C$$

expressing the fact that subjects belonging to the same coalition can have access to all data (– means that no transformation functions are performed before access is given). L'esempio sotto dovrebbe parare di coalizione e coalition agreement no?

Example 4.1. Let us consider an healthcare scenario, with an hospital A storing patients' information such as patient personal information (e.g., social security number, home address, healthcare insurance, etc.), medical journals and medication records. We suppose there are three different roles among the hospital staff: doctor, nurse and administrative staff. They all need to access patient's data in their daily activities, but it would make sense to have specific access rules based on the role, such as:

- a doctor can have access only to medical journals and medication records;
- a nurse should only have read access on medication records;
- an administrative must have access only to patient's personal information.

In our model, we can specify the three roles as subject's attribute *role* which can have only one of three values `doctor`, `nurse`, and `admin`. Then, we can express the three policy rules as:

$$policy\_rule_{PatientData}(s, \texttt{any}, env, -) \equiv s.role = \texttt{doctor}$$

$$policy\_rule_{PatientData}(s, \texttt{READ}, env, \texttt{medication only}) \equiv s.role = \texttt{nurse}$$

4.2 Data Annotation

In cloud scenarios, it is common to apply metadata tags to resources to perform more sophisticated filtering or reporting on resources. A tag is a label consisting of a key and an optional value that is assigned to a resource. In our case, we exploit tagging to specify attributes of subjects, resources or contextual information. Using data annotation to enforce data protection policies has three major advantages:

1) Metadata tagging permits to extract value also out of possibly unstructured ingested raw data. For example, it is possible to capture document semantics through tags. Thus, tagging helps in defining policies at different levels of granularity, also in case of unstructured data.
2) Tagging enables to categorize resources in different ways (e.g, by purpose, owner, environment, or other criteria), encompassing conventional database schema information. Even relationships between attributes of different data sets can be indicated as tags, without the need of, computationally expensive, data normalization.
3) Tags can also be used to express access control related attributes, such as data sensitivity or multi-level security policies by means of owner-defined security levels, or role-based policies.

When using tagging for access control in a distributed scenario, it is important to devise a consistent set of tag keys, that is, a common vocabulary. However, which tags to apply to which resources differs depending on the specific use case, working environment, or context. Table **??** provides an example of a common vocabulary at the basis of tag-based access control policies. For instance, use case *privacy classification* should be used every time data privacy must be protected, that is, every time personal data are involved. Data privacy generally refers to the ability of a person to determine when, how, and to what extent her personal information can be shared with or communicated to others. According to the General Data Protection Regulation (GDPR) [5], personal data are any information relating to an identified or identifiable natural person (called data subject). Based on this definition and on the technological advances that have improved data collection and analytics, personal data (Personally Identifiable Information - PII) fall into three categories: *i)* explicit identifiers, any information that directly identifies individuals with certainty, such as national ID, assurance number, phone number, passport number; *ii)* quasi identifiers, a set of data attributes that could jointly or uniquely identify a subject when combined with publicly available data, such as ZIP code, date of birth, and address; *iii)* sensitive information, data related to a person without directly identifying her but that, if linked to an individual, reveals sensitive aspects of her private life (e.g., religion belief, health information, legal issues).

4.3 Data Transformation

4.4 Access Control Enforcement

Figure 2 shows how the access control model presented in section 4.1 is integrated in the big data processing system of Figure 1.

Data lineage is a set of complex relationships between datasets D_i and jobs J_i in a pipeline. The picture shows how policy enforcement works along the data lineage obtaining

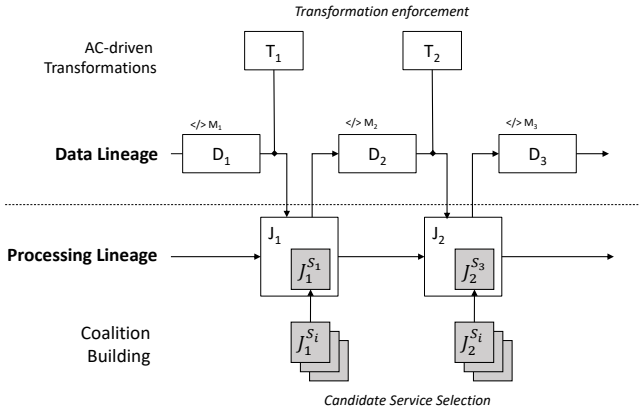

Fig. 2. Access Control Enforcement process.

data protection by means of AC and transformations prior to feed data to a processing job. Along the pipeline, with data flowing from data sources to computation, access control enforcement works as follows:

1) When D_i has to be processed by job J_i provided by a service provider S_j, an access decision has to be taken.
2) A matching rule $policy\_rule_{D_i}(S_j, J_i, env)$ is searched in the policy, considering all the actual values of the attributes of both subjects and objects. If the rule is found, the corresponding data transformation T_1 is performed.
3) Job J_i is executed on D_i after the transformation, generating D_{i+1} that go through the same process before being fed to job J_{i+1}.

5 Architectural Deployment

We present the architectural deployment of our access control system discussing two possible approaches: *i)* centralized deployment; *ii)* decentralized deployment. The choice of the specific deployment has an impact on the way in which the coalition C of organizations o_i is formed as discussed in the following of this section.

5.1 Centralized Deployment

A centralized deployment implements a service orchestration, where an orchestrator Orc provides access control functionalities and mediates access from organization o_i in C to a given dataset D. A service orchestration can be formally defined as follows.

Definition 5.1. Given a big data analytics pipeline $G(V,E)$ in Definition 2.1, a coalition C of organizations $o_i \in O$ each implementing a job J_i, a dataset D, and an orchestrator Orc, a service orchestration is a direct acyclic graph $G^c(V^c,E^c)$, where $V^c=V_I \cup Orc$ and $E^c=\{v_i,Orc\} \cup \{Orc,v_i\}$, with $v_i \in V^c$ modeling a job J_i.

We note that each communication between two subsequent jobs J_{i-1} and J_i in C is mediated by the orchestrator, which enforces all applicable policies. We also note that vertices v_c and v_m of an alternative structure, as well as v_f and v_j of a parallel structure, are included in the orchestrator Orc. Two enforcement processes are implemented in the centralized deployment as follows.

1) **Incoming enforcement.** Policy enforcement is done by the orchestrator before dataset D is released to a specific (set of) job J_i. It is then executed for each edge $\{Orc,J_i\}$ and decrease or maintain the utility of the dataset (i.e., the enforcement has no impact on the dataset or remove some information).
2) **Outgoing enforcement.** Policy enforcement is done by the orchestrator after a resulting dataset D is returned by a specific (set of) job J_i. It is then executed for each edge $\{J_i,Orc\}$ and aims to restore those data that were manipulated before access by J_{i-1}. In other words, once the resulting dataset is returned by J_{i-1}, orchestrator Orc restore those data that were not accessible by J_{i-1} (e.g., by deanonymizing it).

We note that centralized deployment maximizes the utility of the dataset providing each job with the largest amount of data possible, meaning that data transformation assumes a non-monotonic behavior. It is a single point of failure and assumes all jobs to coexist in a single environment. Generalization to multiple environments is possible but outside the scope of this paper.

5.2 Decentralized Deployment

A decentralized deployment implements a service choreography, where organization o_i in C are directly connected and exchange data. A service choreography can be formally defined as follows.

Definition 5.2. Given a big data analytics pipeline $G(V,E)$ in Definition 2.1, a coalition C of organizations $o_i \in O$ each implementing a job J_i, and a dataset D, a service choreography is a direct acyclic graph $G^d(V^d,E^d)$, where $V^d=V$ and $E^d=\{v_i,v_j\}$ with $v_i \in V^d$ modeling a job J_i.

We note that each communication between two subsequent jobs J_{i-1} and J_i in C is direct with no mediation. Each job is complemented with an external plugin enforcing all applicable policies. Policy enforcement is done by the plugin of J_{i-1} before dataset D is released to job J_i. It is then executed for each edge $\{J_{i-1},J_i\}$, enforcing all applicable policies and decrease or maintain the utility of the dataset (i.e., the enforcement has no impact on the dataset or remove some information). We note that decentralized deployment provides a data transformation assuming a not-increasing monotonic behavior. Communications are distributed among different job platforms requiring data transfer during analytics. This choice could decrease performance when huge datasets must be distributed.

6 Coalition Building

Coalition building is a crucial process having direct impact on the quality of the analytics results. Figure 2 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.

Fig. 3. Centralized (a) and decentralized (b) deployment

Let us consider job $J_1^{o_1}$ of Figure 2 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 [10] 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.

6.2 Metrica Composta tipo vikor

6.3 Metrica Composta vikor wighted (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. Pseudo-code for the greedy algorithm is presented in Algorithm 1.

```
GreedyHeuristic(Set S):
1. Initialize an empty set R to store the
    selected nodes
2. While S is not empty:
    a. Calculate the metric for each
        node in S
    b. Select the node with the minimal
        metric and add it to R
    c. Remove the selected node from S
3. Return R
```

6.7 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.

```
SlidingWindowHeuristic(Seq S, int k):
1. Initialize an empty list R to store the
    selected nodes
2. For i from 0 to the length of S - k:
    a. Find the node with the minimal metric
        in the window from i to i+k
    b. Add the selected node to R
3. Return R
```

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.

7 Experiments

8 Related Work

- We believe that the closest approach to the one in this paper is the work of Hu et al. [8], 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.
- [11] 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.

- [6] 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.
- [9] 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] Aissa, M.M.B., Sfaxi, L., Robbana, R.: DECIDE: A New Decisional Big Data Methodology for a Better Data Governance. In: Proc. of EMCIS, vol. 402. Dubai, EAU (2020)

[2] Al-Badi, A., Tarhini, A., Khan, A.I.: Exploring Big Data Governance Frameworks. Procedia Computer Science **141**, 271–277 (2018)

[3] Anisetti, M., Ardagna, C.A., Braghin, C., Damiani, E., Polimeno, A., Balestrucci, A.: Dynamic and Scalable Enforcement of Access Control Policies for Big Data, pp. 71–78. Association for Computing Machinery, New York, NY, USA (2021)

[4] Demchenko, Y., Grosso, P., de Laat, C., Membrey, P.: Addressing big data issues in scientific data infrastructure. In: Proceedings of IEEE CTS. Brisbane, Australia (2013)

[5] EU: The General Data Protection Regulation (GDPR) - Regulation (EU) 2016/679. URL https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX%3A02016R0679-20160504&qid=1532348683434

[6] 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)

[7] Hill, R.C., Lockhartr, H.: eXtensible Access Control Markup Language (XACML) Version 3.0. OASIS Standard (2013). URL https://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-os-en.html

[8] 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)

[9] 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)

[10] 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

[11] 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)