Published version available at https://doi.org/10.1007/s00778-021-00709-x

Noname manuscript No.
(will be inserted by the editor)

An authorization model for query execution in the cloud

Sabrina De Capitani di Vimercati · Sara Foresti · Sushil Jajodia · Giovanni Livraga · Stefano Paraboschi · Pierangela Samarati

Received: date / Accepted: date

Abstract We present a novel approach for the specification and enforcement of authorizations that enables controlled data sharing for collaborative queries in the cloud. Data authorities can establish authorizations regulating access to their data distinguishing three visibility levels (no visibility, encrypted visibility, and plaintext visibility). Authorizations are enforced accounting for the information content carried in the computation to ensure no information is improperly leaked and adjusting visibility of data on-the-fly. Assignment of operations to subjects takes into consideration the cost of operation execution as well as of the encryption/decryption operations needed to make the assignment authorized. Our approach enables users and data authorities to fully enjoy the benefits and economic savings of the competitive open cloud market, while maintaining control over data.

Keywords Authorization model · Collaborative query evaluation · Plaintext and encrypted visibility · Implicit attributes · Equivalent attributes · Relation profile

This work was supported in part by the Office of Naval Research under grant N00014-20-1-2407, by the Army Research Office under grant W911NF-13-1-0421, by the National Science Foundation under grant CNS-1822094, by the EC within the H2020 Program under grants 825333 and 101017171, and by JPMorgan Chase & Co.

S. De Capitani di Vimercati, S. Foresti, G. Livraga, P. Samarati
Università degli Studi di Milano, Italy
E-mail: {firstname.lastname}@unimi.it

S. Jajodia
George Mason University, USA
E-mail: jajodia@gmu.edu

S. Paraboschi
Università degli Studi di Bergamo, Italy
E-mail: parabosc@unibg.it

1 Introduction

Today's ICT (Information and Communication Technology) scenarios are seeing an ever-growing explosion of data collection, sharing, and collaborative processing, as well as an ever-increasing need to efficiently perform extensive data analysis tasks over data produced and controlled by different parties (e.g., in medical or genomic data applications). The evolution of technology, and especially of the cloud computing paradigm, offering a variety of storage and computation providers with different costs and performance guarantees, well responds to such demands and needs. Multi-provider applications can leverage the richness and diversity of the cloud market by involving different parties depending on specific needs and economic benefit. Users and companies can then enjoy clear social and economic benefits in terms of convenient, scalable, and elastic availability of services. At the same time, however, data could be sensitive, proprietary, or simply subject to access restrictions that can affect the possibility of relying on external cloud providers for their management and processing [30]. Addressing security concerns over data exposure by restricting processing within the premises of each individual data authority (i.e., the entity controlling the data) or at the user side, can hinder the ability to fully benefit from the rich and diverse cloud market offering. This represents a significant barrier towards the evolution of the market and the related economic growth.

In this paper, we address this problem and propose a novel approach enabling collaborative and distributed query execution with the controlled involvement of cloud providers that might be not fully trusted to access the data content. Our goal is twofold: first, to allow data authorities to make their data available

for possible collaborative processing, while maintaining control over them; second, to allow users accessing such data to leverage the rich and diverse offer of the cloud market, by relying on cloud providers for performing queries over such data.

The core of our proposal is a simple, yet flexible, authorization model that enjoys the great advantage of simplicity of specification and management. Each data authority can establish authorizations regulating the release of data under its control to other subjects (i.e., users, providers, and other data authorities). Authorizations are specified by each authority independently (no cross-domain authorization or collaborative administration is required) and selectively grant visibility on the data to other subjects. Visibility can be granted either plaintext or encrypted. Subjects authorized for encrypted visibility over some data can perform computations (e.g., evaluate conditions or perform joins) over the data without accessing the actual data values. Leveraging the availability of solutions that support operations on encrypted data (e.g., CryptDB [23] and the SEEED framework over the SAP Hana DBMS [16]), this feature increases the spectrum of potential providers to which operations within a query can be assigned. Query execution can then selectively involve, in the different steps of the computation, different data authorities and cloud providers as deemed desirable for economic or performance reasons. Encryption/decryption operations are injected in the query process and enforced on-the-fly as needed to disable/enable data visibility as demanded by authorizations and operation requirements. Authorization enforcement entails controlling not only direct data access, or release, but also accounting for information implicitly conveyed as a result of a computation.

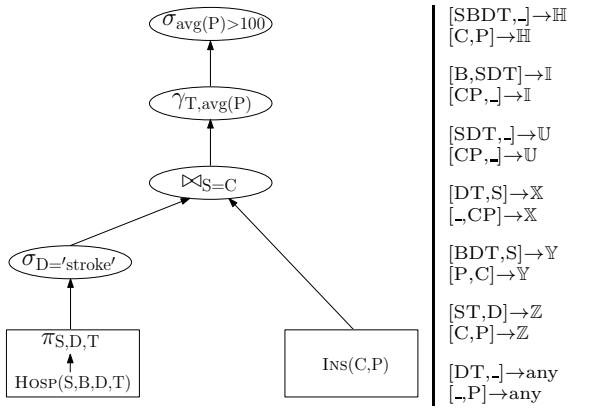
Running example. For concreteness, but without loss of generality, we frame our work in the context of relational database systems. We consider queries of the general form "SELECT FROM WHERE GROUP BY HAVING" that can include joins among distinct relations under control of different data authorities. We also support renaming operations on attributes and queries that combine the results of other queries of the general form above through set operators (i.e., union, intersection, difference). Execution of queries is performed according to a query plan established by the query optimizer. The query plan is represented as a tree `T(N)`, with `N` the set of nodes in the tree, whose leaves are base relations and whose non-leaf nodes are operations to be executed to perform the query. We assume the query plan to be produced with classical optimization criteria and, in particular, we assume that projections and selections are pushed down to avoid retrieving data that are not of interest for the query. Graphically, we represent a leaf node as a square box that contains (the projection of) a source relation. We refer to leaf nodes as base relations. In this paper, we consider as a running example two data authorities: a hospital $\mathbb{H}$, storing relation Hosp(S,B,D,T), reporting SSN, Birth, Disease, and Treatment of hospitalized patients; and an insurance company $\mathbb{I}$ storing relation Ins(C,P), reporting, for each Customer, the insurance Premium. We assume a user $\mathbb{U}$ who submits a query, and three cloud providers $\mathbb{X}$, $\mathbb{Y}$, $\mathbb{Z}$ offering computational power. Our running example considers the execution, on behalf of user $\mathbb{U}$, of query "SELECT T, avg(P) FROM Hosp JOIN Ins ON S=C WHERE D='stroke' GROUP BY T HAVING avg(P)>100" retrieving, for each treatment given to patients hospitalized for stroke, the average insurance premium (if greater than US$ 100). Figure 1(a) illustrates a query plan for our query. For simplicity, in the figure and in the remainder of this paper, we denote a set of attributes simply with the sequence of the attributes composing it, omitting the curly brackets and commas (e.g., SBDT stands for {S,B,D,T}).

Outline. The remainder of this paper is organized as follows. Section 2 presents our authorization model. Section 3 describes the concept of relation profile, capturing the informative content of a relation. Section 4 shows how protection requirements stated by authorizations must be considered to ensure that data are properly protected in query execution. Section 5 describes the use of on-the-fly encryption and decryption for protecting data in a computation, based on the assignment of query operations to subjects. Section 6 shows how to compute an assignment that enjoys minimum cost. Section 7 illustrates key management and query dispatch to the subjects involved in a query execution. Section 8 presents experimental results. Section 9 discusses related work. Finally, Section 10 concludes the paper. The proofs of theorems can be found in Appendix A.

2 Authorization model

We assume a simple, yet expressive, authorization model in which each data authority specifies authorizations regulating the release of its data. Authorizations are defined at the fine-grained level of attribute specifying, for each attribute, whether a *subject* (i.e., a user, a data authority, or a cloud provider) can have:

- *plaintext visibility*: the subject has complete visibility on the values of the attribute;
- *encrypted visibility*: the subject cannot view the plaintext values of the attribute, but can view an encrypted version of them;



(a) Query plan (b) Authorizations

Fig. 1 An example of a query plan (a) and of authorizations on relations Hosp and Ins (b)

– *no visibility*: the subject cannot view the values of the attribute at all (neither plaintext nor encrypted).

While plaintext and no visibility do not require explanation, since they correspond to traditional ways of regulating access, the encrypted visibility, which represents a characteristic and strength of our proposal, deserves some clarification. The reason behind the consideration of the encrypted visibility is to provide a subject with the ability to operate on an attribute for performing joins with other relations or for evaluating conditions on encrypted values (supported by the kind of encryption used), while not releasing to the subject the actual values of the attribute. In the authorization model, we do not distinguish among different encryption schemes, so to leave the model simple and the approach flexible. In fact, expressing the encryption scheme in the authorizations would introduce considerable complexity in the specifications, without providing an actual advantage in the end. As the experience of null values shows, it is important to maintain specifications simple and intuitive (the introduction of multiple null values in SQL-92 was quickly deprecated). The distinction among encryption schemes will be made by the query optimizer in the generation of the query plan, depending also on the operations that the query performs on the encrypted data (Section 7).

Consistently with standard security practice, we assume a "closed" policy for the specification of authorizations, meaning that only accesses explicitly authorized are allowed (i.e., 'no visibility' does not need to be specified, as it applies whenever the other two do not). Authorizations are then defined as follows.

Definition 1 (Authorization) Let R be a relation and $\mathcal{S}$ be a set of subjects. An *authorization* is a rule of the form $[P, E]\rightarrow S$, where $P\subseteq R$ and $E\subseteq R$ are subsets of attributes in R such that $P\cap E=\emptyset$, and $S\in\mathcal{S}\cup\{\text{any}\}$.

Authorization $[P, E]\rightarrow S$ states that subject S can view attributes P in plaintext and attributes E encrypted. Sets P and E are required to be disjoint. However, we note that an authorization that permits a subject S to access an attribute a in plaintext also allows S to access the encrypted version of the attribute. We assume that, for each relation, a subject can hold at most one authorization (the consideration of multiple authorizations would not increase expressivity). Since the set of subjects who might be involved in a query, and for whom release of data may be requested, may not be completely known a priori, a default authorization can be specified, which applies to all subjects for which no explicit authorization already exists for the interested relation. This is accommodated by the consideration of value 'any' as subject of the authorization.

We expect users to have authorizations that include plaintext attributes only, since users need to be able to access the queries' responses and manage keys for attributes encrypted in the computation. We also expect the data authority storing a relation to hold an authorization for accessing its content in plaintext (i.e., S storing $R(a_1, \ldots, a_n)$ holds authorization $[\{a_1, \ldots, a_n\}, \_]\rightarrow S$). Cloud providers and other data authorities may instead have authorizations that also include encrypted attributes, allowing them to operate on these attributes without viewing their plaintext values. Figure 1(b) illustrates an example of authorizations for our running example.

3 Relation content model

To determine whether the release of a relation to a subject should be accepted according to authorizations, we introduce the concept of *relation profile* capturing the informative content of a, base or derived (i.e., computed by a query), relation. In the following, we first illustrate how attributes that do not belong to a relation schema can influence the definition of its profile, and then formally define relation profiles.

3.1 Implicit, equivalent, and renamed attributes

A relation resulting from a computation can convey information on attributes not explicitly appearing in its schema. This may happen due to the evaluation of a selection condition, of a rename or grouping operation, or of a user defined function (udf) on attributes that are then removed from the relation schema through a

projection. As a simple example, the relation resulting from "SELECT A FROM R WHERE B='10'", while containing only A in its schema, indirectly leaks information on the values of attribute B as well, and should therefore not be visible to subjects not authorized to see either A or B. A similar observation holds for the relation resulting from "SELECT A FROM R_1 JOIN R_2 ON A=B" which, while including only attribute A in its schema, conveys also information on B, as A and B satisfy the equality predicate (hence, granting visibility on A implies leaking also B). Similarly, the relation resulting from "SELECT B AS A FROM R", while including only A in its schema, releases the values of attribute B, hence the relation should be visible only to subjects authorized for B. Note that, in this case, authorization control cannot be performed against A itself, since it is not in the schema of any base relation.

Capturing the informative content of a relation R (resulting from a computation) requires then to take into account such indirect information leakage and relationships among attributes, which we characterize through the concepts of *implicit*, *equivalent*, and *renamed* attributes.

Implicit attributes. Implicit attributes are attributes not necessarily appearing in a relation schema but that have been taken into account in the computation of the relation. Basically, implicit attributes for a relation R are all those attributes that appear in a selection condition or grouping operation in the (sub-)query producing R. The information indirectly conveyed differs depending on the selection condition considered. For instance, a selection condition 'B=10' leaks the fact that all the tuples in the result have value of B equal to 10, disclosing B precisely even if it is not explicitly visible in the relation. A selection condition 'B>10' leaks instead the fact that the tuples appearing in the relation have a value for B greater than 10, but without leaking B's actual values. The evaluation of a GROUP BY clause over B is similar to the evaluation of equality condition 'B=*value*', where *value* may be unknown. Consistently with the fact that we operate at the schema level, we do not distinguish among the degrees of leakage and assume an attribute to be *implicitly visible* in a relation (i.e., indirectly exposed) if the attribute was taken into account – in some way – in the computation of the relation. The concept of implicit visibility applies to both plaintext and encrypted attributes.

Equivalent attributes. Equivalence among attributes captures the fact that some attributes have been connected in a computation (i.e., some conditions among them have been applied) and therefore visibility of one attribute indirectly leaks the other(s). Like for implicit attributes, the degree of such a leakage can depend on the condition enforced. For instance, condition 'A=B' implies precise leakage of the values of B from the visibility of A, while condition 'A>B' entails a partial leakage, as a subject viewing A can only infer the fact that B has a value lower than the one visible for A. Again, we do not distinguish among different degrees of leakage (which would introduce considerable complexity and fuzziness in the approach, with limited advantages in the enforcement of authorizations), but simply capture such a connection between the attributes, considering them as equivalent from the point of view of authorization enforcement (as visibility of one entails some visibility of the other). Given a relation R, we say that two attributes are *equivalent* if the (sub-)query producing R involves a condition comparing them. The equivalence relationship is symmetric and transitive. Different sets of equivalent attributes can exist for a given relation. The equivalence relationship can apply to both explicit as well as implicit attributes, and to plaintext as well as encrypted attributes.

Renamed attributes. Renamed attributes are attributes that do not appear in the schema of base relations as they result from a change in the name of original attributes through a rename operation. The release of a relation with a renamed attribute clearly discloses the original attribute, even if such attribute does not appear in the relation schema. For instance, query "SELECT B AS A FROM R" reveals the values of attribute B under attribute name A, but no authorization regulates the release of A. Clearly, the authorizations originally defined over B must apply also to A, since B is just a different name for A. We refer to A as the renamed version of B. The concept of renamed attribute applies to both explicit as well as implicit attributes, and to plaintext as well as encrypted attributes.

3.2 Relation profile

We are now ready to define the profile of a relation, capturing the informative content carried by the relation in terms of attributes explicitly as well as implicitly visible and taking into account information conveyed by equivalent and renamed attributes. In the following, we refer to attributes explicitly visible in a relation as *visible* attributes, and to those implicitly leaked as *implicit*. In addition, attributes can be *plaintext* or *encrypted*.

Definition 2 (Relation Profile) Let R be a relation. The *profile* of R is a 6-tuple of the form $[R^{vp}, R^{ve}, R^{ip}, R^{ie}, R^{\simeq}, R^{\lhd}]$ where: R^{vp} and R^{ve} are the *visible* attributes appearing in R's schema in plaintext

(R^{vp}) or encrypted (R^{ve}) form; R^{ip} and R^{ie} are the *implicit* attributes conveyed by R, in plaintext (R^{ip}) or encrypted (R^{ie}) form; $R^{\simeq}$ is a disjoint-set data structure representing the closure of the equivalence relationship implied by attributes connected in R's computation; and $R^{\lhd}$ is a set of attribute pairs $[a', a]$ denoting the renaming of a as a'.

The profile of a base relation has all the elements but R^{vp} empty, since it is assumed accessible in plaintext and does not carry any implicit content or equivalence/renaming relationship. (Note that plaintext accessibility of a relation does not imply that it is stored in plaintext but only that it is accessible in plaintext by the data authority storing it.) Formally, the profile of a base relation $R(a_1, \ldots, a_n)$ is then $[\{a_1, \ldots, a_n\}, \_, \_, \_, \_, \_]$.

The profile of the relation resulting from a query depends on the profile of the operand relations and on the operators involved in its computation. Every operator only operates on visible attributes (i.e., attributes in R^{vp} and R^{ve}, which belong to the schema of the operand relation R), but it may affect also implicit attributes in the profile of the resulting relation. In the following, we illustrate the profile resulting from the application of projection, selection, cartesian product, join, group-by, rename, union, intersection, difference, and udf operators as well as encrypt/decrypt operators. In the treatment, with a slight abuse of notation, we will use symbol $\cup$ to denote the insertion into $R^{\simeq}$ of the equivalence relationship among a set A of attributes. In other words, $R^{\simeq} \cup A$ adds A to $R^{\simeq}$ if no set in $R^{\simeq}$ intersects A; it merges all the sets intersecting A as well as A in a single set in $R^{\simeq}$, otherwise. $R_i^{\simeq} \cup R_j^{\simeq}$ implies inserting into $R_i^{\simeq}$ all the equivalence sets in $R_j^{\simeq}$ (or, equivalently, vice versa). Also, given an attribute a' and component $R^{\lhd}$, function $\omega(a', R^{\lhd})$ returns a if $R^{\lhd}$ includes a pair $[a', a]$; it returns a', otherwise. We use $\omega(A, R)$ to denote the application of function ω to each attribute in A on $R^{\lhd}$. In other words, function ω returns the original names of the attributes on which it applies (if any) or the attributes themselves.

Graphically, we represent the profile of a relation as a tag attached to the relation's node (or the node of the operator producing it in case of a derived relation), with four components: v (visible attributes R^{vp} and R^{ve}), i (implicit attributes R^{ip} and R^{ie}), $\simeq$ (sets of equivalent attributes $R^{\simeq}$), and $\lhd$ (pairs of attributes involved in renaming operations $R^{\lhd}$). Within visible and implicit attributes, we distinguish the encrypted ones (i.e., R^{ve} and R^{ie}) by representing them on a gray background. We represent an encryption operation as a gray box, containing the attributes to be encrypted, on top of the operand relation. We represent a decryption operation as a white box, containing the attributes to be decrypted, below the node representing the operator. Figure 2 illustrates the graphical representation of the profiles resulting from relational and udf operations, reporting, for each operator, the general formula (on the left) and an example (on the right). Similarly, Figure 3 illustrates the graphical representation of the profiles resulting from encryption and decryption operations. For the sake of readability, in the figures and in the paper, we omit the $\lhd$ component in the graphical representation of relation profiles, reporting it only for the rename operator. Indeed, only the rename operator has an effect on component $R^{\lhd}$, while $R^{\lhd}$ of the result is the same as $R^{\lhd}$ of the operand(s) for all the other unary operators and it is the union of them in case of binary operators. We now discuss the profile resulting from the application of each operator.

Projection (π). It returns a subset of the attributes in the schema of its operand. The profile of the resulting relation simply contains, in the visible attributes, only those attributes that have been projected. The implicit and renamed attributes as well as the equivalence sets are the same as the ones of the operand.

Selection (σ). It returns a subset of the tuples of its operand, based on the evaluation of a condition on visible attributes. Since a selection does not have any effect on the schema of the operand relation, the result has the same visible and renamed attributes as the operand. The other components of the profile depend on the kind of condition to be evaluated. For conditions of the form '$a\ op\ x$', with x a value, attribute a is added to the implicit attributes (either encrypted or plaintext, consistently with how a is visible in the operand). For conditions of the form '$a_i\ op\ a_j$', equivalence $\{a_i, a_j\}$ is added to the equivalence set. Note that attributes a_i and a_j must be either both visible plaintext or both visible encrypted for the evaluation of condition '$a_i\ op\ a_j$'.

Cartesian product ($\times$). It returns the cartesian product of two operand relations R_l and R_r, that is, all possible combinations of their tuples. The plaintext/encrypted attributes visible or implicit in the resulting relation, renamed attributes, and the sets of equivalent attributes are then simply the union of the corresponding sets in the profiles of the operands.

Join ($\bowtie$). It returns a relation that contains the concatenation of the tuples of the operands R_l and R_r that satisfy a join condition C, which is a Boolean formula of basic conditions of the form '$a_i\ op\ a_j$'. It is then equivalent to a selection operating on the cartesian product of the operands (i.e., $\sigma_C(R_l \times R_r)$). The profile of the

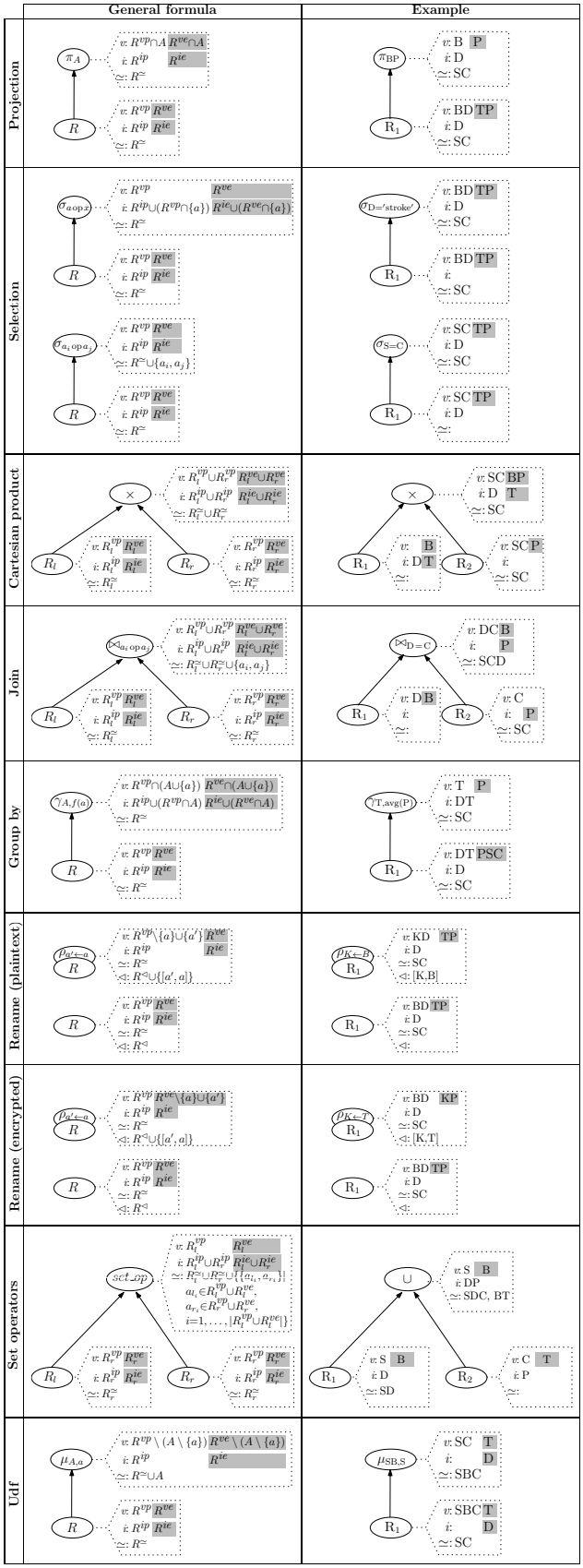


Fig. 2 Graphical representation of the profiles resulting from relational and udf operations

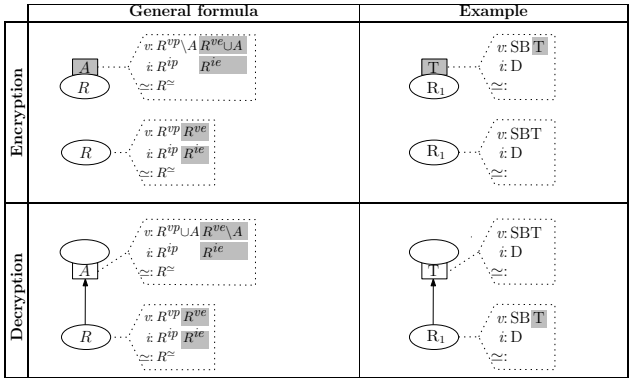


Fig. 3 Graphical representation of the profiles resulting from encryption/decryption operations

result reflects then the information conveyed by both these operators. Also in this case, for each pair $\{a_i,a_j\}$ of attributes appearing together in a condition C, a_i and a_j must be both plaintext or both encrypted for the evaluation of the join condition.

Group by (γ). It groups the operand relation by a given set of (plaintext or encrypted) attributes A, then evaluating an aggregate function f on an attribute a. For simplicity, we consider the attribute resulting from $f(a)$ with the same name as a. The case where $f(a)$ takes a different name can be accommodated with the rename operator. The profile of the resulting relation contains, in the visible attributes, only those attributes on which the grouping (A) and aggregate function (a) operate (when $f(a)$ is COUNT($*$), only attributes in A are maintained). Attributes appearing in the grouping function (A) are added to the implicit attributes (to capture the possible information leakage from their grouping).

Rename (ρ). It changes the name of a subset of the (plaintext or encrypted) visible attributes of its operand. The only effect of the operator is the different name of the renamed attributes in the visible component. The implicit attributes and equivalence sets do not change. For each renamed attribute a' resulting from the application of the rename operator over attribute a, pair $[a', \omega(a, R^{\lhd})]$ is added to the set $R^{\lhd}$ of renamed attributes. Note that the use of function ω in the added pair ensures that the rename component $R^{\lhd}$ keeps always track of the correspondence between a renamed attribute and the corresponding attribute appearing in a base relation, enabling transitive closure of chains of rename operations.

Union, intersection, difference ($\cup, \cap, \setminus$). They are binary operators that apply to operand relations R_l and R_r characterized by the same number of visible attributes, which need to be of compatible domains and

represented in the same form (i.e., the i-th attributes in R_l and R_r must be both plaintext or both encrypted for the evaluation of set operators). Set operators return all the tuples that are in: R_l or R_r (union $\cup$); both R_l and R_r (intersection $\cap$); R_l but not in R_r (difference $\setminus$). The visible attributes of the resulting relation correspond to the visible attributes of the first operand (R_l). The implicit attributes, the sets of equivalent attributes and of the renamed attributes are the union of the corresponding components in the profiles of the operands. The fact that the i-th attribute a_{l_i} appearing in the resulting relation is derived from the i-th attributes a_{l_i} and a_{r_i} in the operand relations R_l and R_r is represented through the addition of a pair $\{a_{l_i}, a_{r_i}\}$ in the equivalence set of the result.

User defined function (μ). It performs a time-consuming procedural computation (e.g., machine learning and data analytics [8]) over the operand relation, elaborating the values of a set A of attributes (all plaintext or encrypted) in its schema. We assume a general udf operator with a set (A) of attributes as input and an attribute (a) as output. For simplicity, we assume the attribute in output to have the same name as one of the attributes in input. The case where the result assumes a different name can be accommodated using the rename operation. The profile of the resulting relation has, as visible attributes, the attribute returned as output together with the visible attributes of the operand on which the udf does not operate. The implicit and renamed attributes are the same as the ones in the operand. The equivalence relationship is obtained from the one in the operand by adding the set of attributes on which the udf operates. This reflects the fact that the attribute in output depends on all the attributes on which the udf has operated.

Encryption. It changes a relation by encrypting some of its plaintext attributes. The result has the same profile as the operand, apart from the fact that the attributes on which encryption is applied are moved from visible plaintext to visible encrypted.

Decryption. It changes a relation by decrypting some of its encrypted attributes. The result has the same profile as the operand, apart from the fact that the attributes on which decryption is applied are moved from visible encrypted to visible plaintext.

While relations in the query plans, and their profiles, can contain renamed attributes, authorizations are defined over attributes appearing in base relations only and do not regulate the release of attributes with different (new) names. To determine whether a relation can be released, we need to reconstruct the correspondence between renamed attributes in its profile and attributes in the base relations. In other words, for each relation R, each component in R's profile must be closed (possibly implying recursively chasing a sequence of rename operations) against the relationships in $R^{\lhd}$. To maintain the notation simple, instead of repeating such a closure throughout the model, we simply (and equivalently) assume profiles to be closed against the renaming relationship so to refer to attributes in the base relations. Formally, the closed profile of a relation is defined as follows.

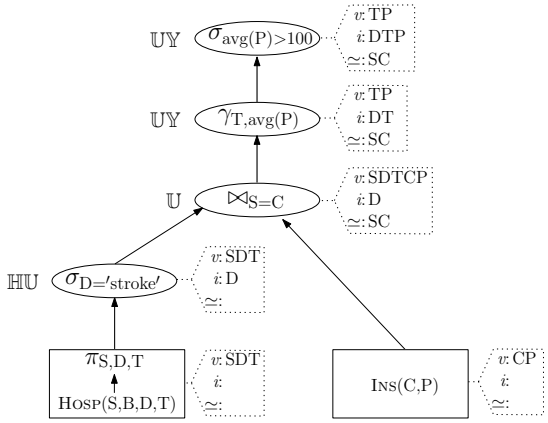


Fig. 4 Query plan with profiles and authorized assignees

Definition 3 (Relation Profile – Closed) Let R be a relation and $[R^{vp}, R^{ve}, R^{\dot{ip}}, R^{ie}, R^{\simeq}, R^{\lhd}]$ be its relation profile (Def. 2). The *closed profile* of R is a 5-tuple of the form $[R^{vp}_{\lhd}, R^{ve}_{\lhd}, R^{\dot{ip}}_{\lhd}, R^{ie}_{\lhd}, R^{\simeq}_{\lhd}]$ where $R^{vp}_{\lhd} = \omega(R^{vp}, R^{\lhd})$, $R^{ve}_{\lhd} = \omega(R^{ve}, R^{\lhd})$, $R^{\dot{ip}}_{\lhd} = \omega(R^{\dot{ip}}, R^{\lhd})$, $R^{ie}_{\lhd} = \omega(R^{ie}, R^{\lhd})$, and $R^{\simeq}_{\lhd} = \omega(R^{\simeq}, R^{\lhd})$.

The closed profile of a relation is equivalent to its profile, since it simply replaces attribute names assigned by rename operations with the corresponding ones in base relations. In the following, we will use the term profile to refer to the closed profile of a relation (Definition 3), and notation $[R^{vp}, R^{ve}, R^{\dot{ip}}, R^{ie}, R^{\simeq}]$ to denote the (closed) profile of R.

Figure 4 illustrates the profiles of the relations resulting from the operations of our running example. Each node has, on its left, the user and a set of cloud providers (we will elaborate on this in the next section). Also, note that there are no encryption/decryption operations, as they do not appear in the original query plan; we will illustrate how and why the query plan is extended with them in Section 5. In the following, given a query plan, we use the term node to denote one of its components (a base relation or an operation) and the term relation to denote either a base relation or the result of an operation (represented by an internal node).

Given a node n_x, representing an operation, R_x denotes the relation resulting from it.

Profiles allow us to capture the informative content of a relation resulting from a computation. The following theorem proves that in a query plan: *i)* attributes appearing in the profile of the relation resulting from an operation survive in the profiles of relations resulting from subsequent operations (i.e., they never disappear from the profile, they can only move from one component to another), and *ii)* equivalence sets can only increase going up in the query plan (i.e., when an attribute is inserted into an equivalence set, it is never removed from it).

Theorem 1 *Let* T(N) *be a query plan.* $\forall n_x, n_y \in$ N *with profile* $[R_x^{vp}, R_x^{ve}, R_x^{ip}, R_x^{ie}, R_x^{\simeq}]$ *and* $[R_y^{vp}, R_y^{ve}, R_y^{ip}, R_y^{ie}, R_y^{\simeq}]$*, respectively, s.t.* n_y *is a descendant of* n_x*:*

i) $(R_y^{vp} \cup R_y^{ve} \cup R_y^{ip} \cup R_y^{ie} \cup \{A \mid A \in R_y^{\simeq}\}) \subseteq (R_x^{vp} \cup R_x^{ve} \cup R_x^{ip} \cup R_x^{ie} \cup \{A \mid A \in R_x^{\simeq}\})$

ii) $\forall A \in R_y^{\simeq} : \exists A' \in R_x^{\simeq}, A \subseteq A'$.

4 Authorized visibility and assignment

The definition of relation profile, capturing the informative content carried by a relation, allows us to regulate query execution ensuring obedience to authorizations. Such regulations concern both visibility of relations as well as execution of operations in the query plan. Since a computation might involve different base relations, different authorization sets (and authorities) might be involved in the control for the release of a derived relation. We will elaborate on this in Section 7. In this section, for simplicity, we assume an overall view of the authorizations and we use notation $\mathcal{P}_S$ ($\mathcal{E}_S$, resp.) as a short-hand for the abstract concept summarizing the attributes that subject S is authorized to access in plaintext (encrypted, resp.) form. In other words, $\mathcal{P}_S = \{a \in P \mid [P, E] \rightarrow S\}$ and $\mathcal{E}_S = \{a \in E \mid [P, E] \rightarrow S\}$. Figure 5 shows the authorizations for our running example and the corresponding overall views for the different subjects.

The following definition captures the authorization control on a relation (based on its profile) to determine whether releasing it to a subject obeys authorizations, taking into account also information leakage caused by implicit attributes and equivalence relationships.

Definition 4 (Authorized Relation) Let R be a relation with profile $[R^{vp}, R^{ve}, R^{ip}, R^{ie}, R^{\simeq}]$. A subject $S \in \mathcal{S}$ is *authorized* for R iff:

1. $R^{vp} \cup R^{ip} \subseteq \mathcal{P}_S$ (authorized for plaintext);
2. $R^{ve} \cup R^{ie} \subseteq \mathcal{P}_S \cup \mathcal{E}_S$ (authorized for encrypted);
3. $\forall A \in R^{\simeq}$, $A \subseteq \mathcal{P}_S$ or $A \subseteq \mathcal{E}_S$ (uniform visibility).

	Authorizations		**Authorized attributes**	
Subject	Hosp(S,B,D,T)	Ins(C,P)	Plaintext	Encrypted
$\mathbb{H}$	[SBDT,_]→$\mathbb{H}$	[C,P]→$\mathbb{H}$	$\mathcal{P}_{\mathbb{H}}$=SBDTC	$\mathcal{E}_{\mathbb{H}}$=P
$\mathbb{I}$	[B,SDT]→$\mathbb{I}$	[CP,_]→$\mathbb{I}$	$\mathcal{P}_{\mathbb{I}}$=BCP	$\mathcal{E}_{\mathbb{I}}$=SDT
$\mathbb{U}$	[SDT,_]→$\mathbb{U}$	[CP,_]→$\mathbb{U}$	$\mathcal{P}_{\mathbb{U}}$=SDTCP	$\mathcal{E}_{\mathbb{U}}$=_
$\mathbb{X}$	[DT,S]→$\mathbb{X}$	[_,CP]→$\mathbb{X}$	$\mathcal{P}_{\mathbb{X}}$=DT	$\mathcal{E}_{\mathbb{X}}$=SCP
$\mathbb{Y}$	[BDT,S]→$\mathbb{Y}$	[P,C]→$\mathbb{Y}$	$\mathcal{P}_{\mathbb{Y}}$=BDTP	$\mathcal{E}_{\mathbb{Y}}$=SC
$\mathbb{Z}$	[ST,D]→$\mathbb{Z}$	[C,P]→$\mathbb{Z}$	$\mathcal{P}_{\mathbb{Z}}$= STC	$\mathcal{E}_{\mathbb{Z}}$=DP
any	[DT,_]→any	[_,P]→any	$\mathcal{P}_{\text{any}}$=DT	$\mathcal{E}_{\text{any}}$=P

Fig. 5 Authorizations and corresponding overall views for the subjects of our running example

According to Definition 4, a subject S is authorized to access a relation R iff all the following three conditions hold: *1)* S is authorized to access in plaintext all the (visible or implicit) attributes represented in plaintext in R; *2)* S is authorized to access in plaintext or in encrypted form all the (visible or implicit) attributes represented in encrypted form in R; *3)* S is authorized to access in the same form (either plaintext or encrypted) all the equivalent attributes, that is, attributes that appear together in an equivalence set in $R^{\simeq}$.

Conditions 1 and 2 correspond to a simple enforcement of authorizations, taking into account both the visible and implicit attributes. Also, Condition 2 considers the fact that subjects authorized for plaintext visibility over an attribute can also have encrypted visibility over the same (since the encrypted representation conveys less information than the plaintext one). Condition 3 enforces control on indirect information leakage caused by equivalence relationships established in query computation, to prevent unauthorized exposure of information. It requires the subject to have the authorizations for the attributes in equivalence sets, since the relation implicitly carries information about them. In other words, since they leave a trace in the computation result, all attributes in equivalence sets are always treated as implicit attributes. Condition 3 also imposes that, within each equivalence set, the authorizations be the same (either plaintext or encrypted) for all attributes in the set. In fact, equivalence relationships in a profile express the fact that some attributes have been related in a computation (e.g., an equi-join operation) and therefore visibility of one attribute in an equivalence set leaks information on the other attributes in the same set. Imposing uniform visibility allows us to account for such inference channels, blocking them when not consistent with the authorizations. Note that uniform visibility must be satisfied for all attributes in an equivalence set, regardless of whether

they belong to the relational schema (i.e., they are visible).

Example 1 Consider the authorizations in Figure 5 and a relation R with profile [P, BSC, _, _, {SC}]:

- $\mathbb{Y}$ is authorized for R;
- $\mathbb{H}$ is not authorized for R (condition 1, attribute P);
- $\mathbb{U}$ is not authorized for R (condition 2, attribute B);
- $\mathbb{I}$ is not authorized for R (condition 3, attributes SC).

Note that the enforcement of uniform visibility entails a possibly counter-intuitive result: a subject could be not authorized for a relation due to the subject's plaintext visibility over some attributes, while another subject that, on these attributes, has only encrypted visibility could be authorized for the relation. For instance, with reference to Example 1, $\mathbb{I}$ is not authorized for R because it has plaintext visibility over C and encrypted visibility over S (and the equivalence between C and S could leak S to $\mathbb{I}$), while $\mathbb{Y}$ is authorized for R since it has only encrypted visibility over C and S, and therefore cannot draw any inference from R.

Definition 4 states when a subject can be authorized for a relation, based on its authorizations and on the relation profile. Another aspect involved in the enforcement of authorizations concerns regulating the assignment of operations within a query plan to authorized subjects. An operation of the query plan, corresponding to a non-leaf node in the tree, operates on one or two operand relations, and produces a relation as output. A subject can be authorized for the execution of an operation if and only if it is authorized for all the relations involved: the operand(s) as well as the result. The authorized visibility for the operand(s) is needed since otherwise the subject could not access them. The authorized visibility for the result enforces the control over the information entailed by the execution of the operation itself. This is captured by the following definition.

Definition 5 (Authorized Assignee) Let $\mathtt{T(N)}$ be a query plan, $n{\in}\mathtt{N}$ be a non-leaf node, n_l,$n_r{\in}\mathtt{N}$ be its children (if any) producing relations R_l and R_r, and $\mathcal{S}$ be a set of subjects. Subject $S{\in}\mathcal{S}$ is an *authorized assignee* of n over R_l and R_r iff S is authorized for R_l, for R_r, and for the relation produced by n, according to Definition 4. Function $\lambda : \mathtt{N} \rightarrow \mathcal{S}$ is said to be an *authorized assignment function* for $\mathtt{T(N)}$ iff $\forall n \in \mathtt{N}$, $\lambda(n)$ is an authorized assignee of n.

Subjects appearing on the left-hand side of each node in Figure 4 are authorized assignees for the node. Leaf nodes do not have any assignee since they remain with the party storing the corresponding base relation.

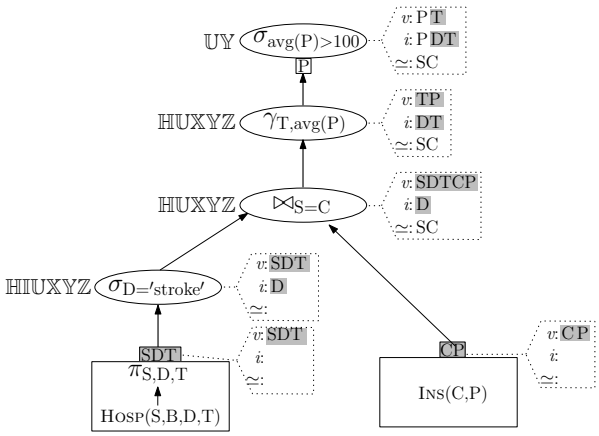


Fig. 6 An extended query plan

5 Extended plans and encryption/decryption

Given a query plan, our goal is to produce an authorized assignment of operations to subjects. While the definitions in Section 4 accounted for the possible presence of encrypted attributes, the original query plan, including only operations requested by the query computation, does not include any encryption/decryption operation. Encryption and decryption operations are inserted on-the-fly by our approach to adjust visibility of attributes as required by operation requirements or authorizations. Encryption protects attributes so to permit the assignment of operations to subjects that could not be considered otherwise. Decryption permits accessing plaintext values of encrypted attributes when needed in the computation. For instance, assume that, for the query plan in Figure 4, all operations but the final selection ($\sigma_{\text{avg(P)}>100}$) could be performed on encrypted values. If all attributes were encrypted at their source (and avg(P) decrypted only for the last operation), more subjects could be considered for executing operations in the query. Figure 6 illustrates the query plan in Figure 4 extended to consider such encryption and decryption operations, reporting on the left-hand side of each node the subjects that could now be considered for the execution of the node's operation. The specific encryption scheme to apply for the encryption of each attribute is decided by the query optimizer in the analysis of the query plan, depending on the kind of operations to be supported over such attributes (Section 7). For instance, deterministic symmetric encryption can be used to efficiently support evaluation of equality conditions in joins and selections, while not disclosing plaintext data values.

A query plan $\mathtt{T}'$ that is obtained by inserting encryption and decryption operations into another query plan

T is called an *extended query plan* for T and is defined as follows.

Definition 6 (Extended Query Plan) Let T(N) be a query plan. A query plan $\mathtt{T}'(\mathtt{N})$ is an *extended query plan* for T iff $\mathtt{T}'$ is T enriched with some encryption and decryption operations.

In the following, the set of extended query plans for T is denoted $\mathcal{T}$. As said, encrypting attributes enables the consideration, for the assignment of an operation, of subjects not otherwise authorized for the execution of the operation. However, the encryption needed to make assignments authorized eventually depends on the actual subjects to which operations are assigned (e.g., P would need to be encrypted for assigning the execution of the join to $\mathbb{X}$, but could remain in plaintext if the join is assigned to $\mathbb{Y}$). There are basically two opposite approaches that can be followed in the insertion of encryption/decryption operations in the query plan, corresponding to maximizing or minimizing visibility of attributes. Maximizing visibility corresponds to always leave plaintext visibility of data, applying encryption only when strictly needed for protecting attributes visibility from the subject executing a specific operation. Minimizing visibility corresponds to always apply encryption by default, decrypting attributes only when needed for operation execution. Each of the two extremes has some pros and cons. Maximizing visibility by default can avoid unnecessary encryption/decryption operations and allows for operating as much as possible on plaintext data, but could reduce the number of subjects to which an operation can be assigned. For instance, suppose that attribute D is not encrypted for the execution of the selection operation ($\sigma_{\mathrm{D}='\mathrm{stroke}'}$), since such an operation is assigned to $\mathbb{H}$, which can see D in plaintext. Then, provider $\mathbb{Z}$ cannot be considered for the join since it does not have the authorization for plaintext visibility of D. In fact, encrypting D only for the join would not protect it from the possible leakage caused by the prior evaluation of the condition (as a matter of fact, D would remain in the implicit plaintext component of the profile of all relations computed after the selection over plaintext attribute D). Maximizing visibility of attributes at a given step may then rule out the consideration of possible subjects in subsequent steps of the query plan. Minimizing visibility, while not affecting the choice of subjects for subsequent operations in the query plan, could result in executing more encryption/decryption operations than actually needed. For instance, encrypting D before the execution of the selection operation may eventually result unnecessary if $\mathbb{Z}$ were not the best choice for the join anyway, implying an overhead for query execution (encryption and possible less efficient evaluation of the condition) which could have been avoided.

To avoid predetermining one of the possible scenarios above, we adopt a more flexible approach by first determining the candidate subjects for the operations in the plan, and then injecting encryption and decryption only as needed, depending on the decided assignment of operations to subjects. The query optimizer can then decide assignments of operations based on costs and performance aspects. Of course, assignment of operations to subjects must be bounded by the authorizations and the operation requirements, which can limit the application of encryption (as some operations need to access some attributes in plaintext for execution). With respect to authorizations, for example, while it is desirable for the execution of the join operation to possibly consider $\mathbb{X}$ (since S and C could be encrypted for that), it does not make sense to consider $\mathbb{I}$ since, as already noted, its non-uniform visibility over S and C (it is authorized to view C in plaintext but S only in encrypted form) rules it out from consideration (Condition 3 of Definition 4). With respect to operation requirements, an attribute should not be encrypted if the operation to be executed on it requires accessing the attribute's plaintext values. For instance, if the encryption scheme available for P does not support range conditions, the possibility of encrypting avg(P) for assigning the last selection operation should be excluded. For operations that are not supported by cryptographic techniques (not existing or not available to the application), we assume the optimizer to specify the need for maintaining data in plaintext for execution of the operation. For each node we then have a set A_p of attributes that are needed in plaintext.

To define the potential candidates for an operation, we first need to characterize the operation requirements, which may limit the application of encryption. We capture this by defining the *minimal visibility* needed over an operand to allow the evaluation of an operator. For instance, in our running example, we assume that the execution of the last selection in the query plan needs to view avg(P) in plaintext, while all other attributes can be encrypted. Intuitively, the minimum required view over an operand for the execution of an operation is the operand relation where all the (visible) attributes, except those that need to be in plaintext for operation execution, are encrypted. This is formally captured by the following definition.

Definition 7 (Minimum Required View) Let T(N) be a query plan, $n \in \mathtt{N}$ be a non-leaf node, n_y be one of its children, producing relation R_y, and A_p be the set of attributes of R_y that must be in plaintext for the execution of n. The *minimum re-*

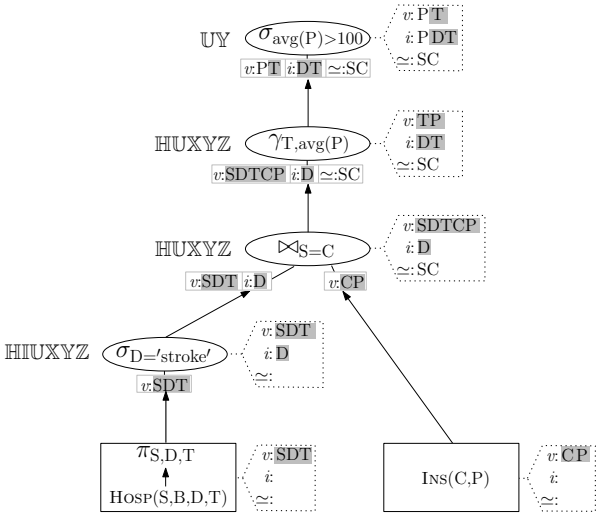


Fig. 7 Minimum required views and assignment candidates

quired view over n_y for the execution of n is relation $\hat{R}_y$=$\texttt{decrypt}(A_p,\texttt{encrypt}(R_y^{vp} \setminus A_p, R_y))$.

In the definition above and in the following, $\texttt{encrypt}(A, R)$ denotes the encryption of attributes A in R and $\texttt{decrypt}(A, R)$ denotes the decryption of attributes A in R. Figure 7 illustrates (in boxes on the arcs from the operands to the operations) the profiles of the minimum required views for our running example. The profiles associated with nodes are those that result assuming as operands such minimum required views. For instance, the minimum required view over Ins for the execution of the join has all attributes (CP) visible and encrypted.

Minimum required views allow us to take into account the visibility requirements for operation execution: only subjects authorized for the minimum required views can be *candidates for the assignment* (since for them the operands can be protected with encryption without affecting operation execution). This is captured by the following definition.

Definition 8 (Assignment Candidates) Let $\texttt{T(N)}$ be a query plan, $n{\in}\texttt{N}$ be a non-leaf node, n_l,$n_r{\in}\texttt{N}$ be its children (if any), and $\mathcal{S}$ be a set of subjects. A subject $S{\in}\mathcal{S}$ is a *candidate* for the execution of n iff S is an authorized assignee of n over $\hat{R}_l$ and $\hat{R}_r$ according to Definition 5. *Candidate assignment function* $\Lambda : \texttt{N} \to 2^{\mathcal{S}}$ associates with each $n \in \texttt{N}$ the set of candidates for the execution of n.

Figure 7 illustrates assignment candidates for the operations of our running example.

The set of candidates along a query plan $\texttt{T(N)}$ enjoys a nice monotonic behavior. For each $n \in \texttt{N}$, the set of candidates of n's ancestors is a subset of the set of n's candidates. This applies to any node representing an operation that does not need to operate on plaintext attributes or that, doing so, leaves an implicit trace of such attributes (i.e., causes them to be included in the implicit attributes of the result's profile). In fact, all such attributes will also remain implicit plaintext in the profile of the minimum required view of any node n_x ancestor of n, and therefore, by definition, any candidate for n_x is certainly also a candidate for n. This is formalized by the following theorem.

Theorem 2 *Let* $\texttt{T(N)}$ *be a query plan,* $n{\in}\texttt{N}$ *be a non-leaf node* n_l,$n_r{\in}\texttt{N}$ *be its non-leaf children, if any.* $\hat{R}_l^{vp} \cup \hat{R}_r^{vp} \subseteq \hat{R}^{ip} \Longrightarrow \Lambda(n_x) \subseteq \Lambda(n), \forall n_x$ *ancestor of* n.

This monotonic behavior can be easily observed in Figure 7, where the set of candidates for each node decreases going up in the query plan.

Intuitively, the set of candidates for a node are *all and only* those subjects that can be made authorized assignees (Definition 5), assuming to extend the query plan with encryption/decryption operations, as stated by the following theorem.

Theorem 3 *Let* $\texttt{T(N)}$ *be a query plan, and* Λ *be a candidate assignment function for it:*

i) $\forall \texttt{T}' \in \mathcal{T}$, λ, *and* $n \in \texttt{N}$, *if* $\texttt{T}'$ *is an extended query plan for* $\texttt{T}$ *and* λ *is an authorized assignment for* $\texttt{T}'$, *then* $\lambda(n) \in \Lambda(n)$.

ii) $\forall \lambda$, *if* $\forall n \in \texttt{N}, \lambda(n) \in \Lambda(n)$, *then there exists an extended query plan* $\texttt{T}'$ *for* $\texttt{T}$ *such that* λ *is an authorized assignment for* $\texttt{T}'$.

In other words: *i)* any assignment that can be made authorized by inserting some encryption and decryption operations is included in Λ, and *ii)* any assignment included in Λ can be made authorized by inserting some encryption and decryption operations. For instance, Figures 8(a-b) illustrate two extended query plans for our running example, assuming operations allocated to the subject indicated on the left-hand side of each node. For convenience of the reader, sets $\mathcal{P}$ and $\mathcal{E}$ of each subject (copied from Figure 5) are repeated in Figure 8(c). In the plan in Figure 8(a): S, C, and P are encrypted before being passed to $\mathbb{X}$, since $\mathbb{X}$ cannot access them in plaintext. In the plan in Figure 8(b), P is encrypted before being passed to $\mathbb{Z}$, since $\mathbb{Z}$ cannot access it in plaintext, while D is encrypted before executing the selection (i.e., the condition on D will have to be dispatched formulated on encrypted values) so not to leave an implicit plaintext trace in the computation given that $\mathbb{Z}$, executing subsequent steps, cannot access

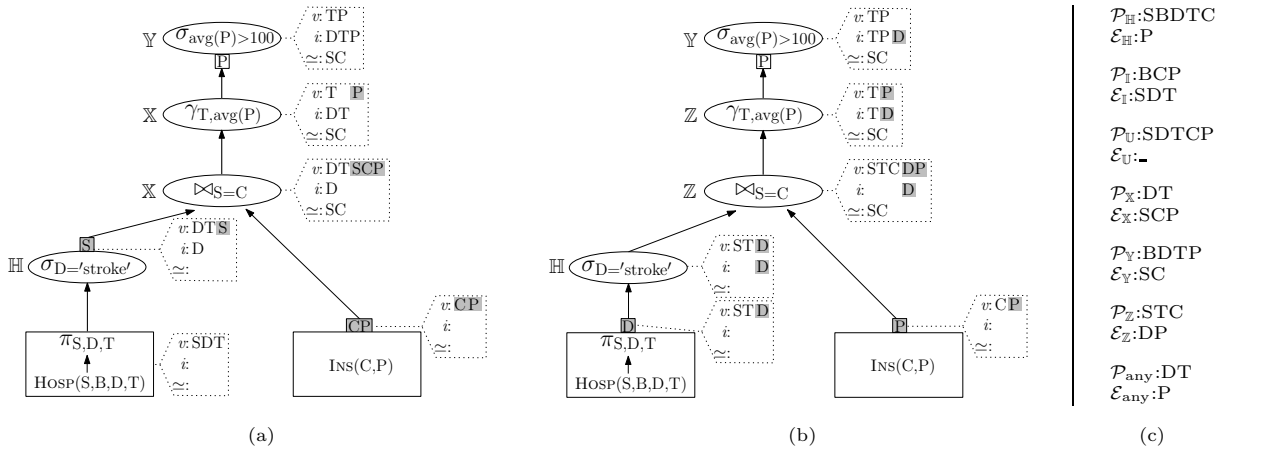


Fig. 8 Extended authorized query plans (a-b) and authorized attributes (c) for the plan in Figure 1

D in plaintext.[1] In both plans, avg(P) is decrypted before the execution of the final selection that needs to access plaintext values. Encryption and decryption operations are assigned to the same subjects as the nodes they complement. Indeed, the subject authorized for a node is also clearly authorized for the preceding decryption (of attributes that are needed in plaintext for the operation) and for the following encryption (of attributes available in plaintext).

Given a query plan $\mathtt{T}$, we are interested in identifying an assignment λ and an extended query plan $\mathtt{T}'$ that makes λ authorized. An extended query plan that makes an assignment authorized is defined as follows.

Definition 9 (Extended Authorized Query Plan) Let $\mathtt{T(N)}$ be a query plan, Λ be a candidate assignment function for it, and λ be a function $\lambda : \mathtt{N} \rightarrow \mathcal{S}$ such that $\forall n \in \mathtt{N} : \lambda(n) \in \Lambda(n)$. An *extended authorized query plan* of $\mathtt{T}$ for λ is an extended query plan $\mathtt{T}' \in \mathcal{T}$ such that λ is an authorized assignment for $\mathtt{T}'$ (Definition 5).

For instance, Figures 8(a-b) illustrate two extended authorized query plans for our running example.

Given a query plan, there are a number of possible authorized assignments in the candidate assignment function Λ. Also, for each possible authorized assignment function λ such that $\forall n \in \mathtt{N} : \lambda(n) \in \Lambda(n)$, there are different ways in which encryption and decryption could be inserted in $\mathtt{T}$ to make λ authorized. For instance, enforcing all encryptions corresponding to the minimum required views (as in Figure 7) could work. Among these extended authorized query plans, the user can choose the one optimizing a parameter of her interest such as cost or performance. In particular, we expect the economic cost to be the driving factor in the choice of the assignment of operations to candidates. Given a query plan $\mathtt{T}$ and a cost function γ, we aim at identifying an assignment λ' and an extended authorized query plan $\mathtt{T}'$ for λ' that minimizes the economic cost $\gamma(\lambda',\mathtt{T}')$ for the evaluation of $\mathtt{T}'$ according to assignment λ'. Formally, our minimization problem is as follows.

Problem 1 (Minimum Cost Assignment) Given a query plan $\mathtt{T(N)}$, a candidate assignment function Λ for it, the set $\mathcal{T}$ of extended query plans for $\mathtt{T}$, and a cost function $\gamma : \Lambda \times \mathcal{T} \rightarrow \mathbb{R}$, determine an assignment function λ' such that $\forall n \in \mathtt{N}, \lambda'(n) \in \Lambda(n)$, and an extended query plan $\mathtt{T}' \in \mathcal{T}$ such that:

1. $\mathtt{T}'$ is an extended authorized query plan of $\mathtt{T}$ for λ';
2. $\forall \lambda''$ such that $\forall n \in \mathtt{N}, \lambda''(n) \in \Lambda(n)$, and $\forall \mathtt{T}'' \in \mathcal{T}$ such that $\mathtt{T}''$ is an extended authorized query plan of $\mathtt{T}$ for λ'', $\gamma(\lambda',\mathtt{T}') \leq \gamma(\lambda'',\mathtt{T}'')$.

The economic cost must clearly take into account the cost of executing computation, the cost of transferring data between different subjects involved in the computation, and the cost of the enforcement of the encryption and decryption operations that need to be injected in the plan. In the next section, we illustrate our approach to model and solve the minimum cost assignment problem.

6 Computing a minimum cost assignment

The results of the previous section prove that, for any operation in the query plan, only subjects in the operation's candidate set need to be considered (Theorem 3, *i*). Also, any of them would do, since any assignment taken from the candidate set can be made

[1] Note that this does not necessarily imply the evaluation of the condition in encrypted form. Since $\mathbb{H}$ is the authority over D and it knows the encryption key (it encrypts D itself), $\mathbb{H}$ can operate on plaintext values and encrypt D afterwards.

authorized by inserting encryption and decryption operations (Theorem 3, *ii*). At the same time, however, each authorized assignment may result in different costs and require injection of different encryption/decryption operations. Also, since encryption and decryption operations themselves bear a cost, the computation of a minimum cost assignment needs to consider also possible encryption/decryption that would be required for the selected candidates to be authorized for the operations assigned to them.

To compute a minimum cost assignment for a query plan T(N), we operate in two steps:

1. compute the candidate assignment function Λ for T(N), that is, identify the set of candidate subjects for the execution of each node in N;
2. determine the assignment λ in Λ (i.e., such that $\forall n \in$ N:$\lambda(n) \in \Lambda(n)$) such that the extended query plan T′ making λ authorized has minimum cost.

The first step restricts the evaluation of possible assignees for each node to the subjects who can be authorized for its execution. The second step determines, among all possible assignments, the one for which the total cost for evaluating the query (including the cost of encryption/decryption operations that need to be injected to make the selected assignees authorized) is minimum.

The computation of candidate assignment function Λ (step 1) is relatively straightforward and can be performed by simply executing a post-order visit of the query plan, also leveraging monotonicity of the candidates along the tree (Theorem 2). The identification of the minimum cost assignment λ (step 2) requires instead solving a minimization problem. We model such a problem as a *binary programming problem* that can be then solved with off-the-shelf solvers. A binary programming problem is formulated as follows: given a *set of variables* that can take values in $\{0, 1\}$, a *set of constraints* over them, and an *objective function*, find an assignment of values to variables that satisfies all the constraints and that minimizes (or maximizes) the value of the objective function. In the remainder of this section, we describe how our optimization problem (Problem 1, step 2 above) can be translated into a binary programming problem by illustrating the corresponding variables, constraints, and objective function.

6.1 Variables

Figure 9 summarizes the variables of our binary programming problem. It distinguishes between *output variables*, which represent the solution to the problem, and *input variables*, which represent its input.

OUTPUT

Variable: 1 if and only if	
$\lambda_{s,n}$:	$\lambda(n)=s$
$vp_{a,n}$:	$a \in R^{vp}$
$ve_{a,n}$:	$a \in R^{ve}$
$ip_{a,n}$:	$a \in R^{ip}$
$ie_{a,n}$:	$a \in R^{ie}$

INPUT

Variable: 1 if and only if	
$\pi_{\bar{n},n}$:	$\bar{n}$ is parent of n
$c_{s,n}$:	$s \in \Lambda(n)$
$\iota_{a,n}$:	n's operation is σ or γ over a
$\hat{eq}_{a,\bar{a},n}$:	n's operation defines an equivalence on a and $\bar{a}$
$auth\_p_{s,a}$:	$a \in \mathcal{P}_s$
$auth\_e_{s,a}$:	$a \in \mathcal{E}_s$
$\hat{vp}_{a,n}$:	a is visible plaintext in n's profile on the MRVs
$\hat{ve}_{a,n}$:	a is visible encrypted in n's profile on the MRVs
$\hat{ip}_{a,n}$:	a is implicit plaintext in n's profile on the MRVs
$\hat{ie}_{a,n}$:	a is implicit encrypted in n's profile on the MRVs

Fig. 9 Variables of the binary programming problem

Output variables. They model the solution of our minimization problem. In particular, given a query plan T(N), the output variables model an authorized assignment function λ for T(N), and the format (plaintext or encrypted) of the, visible or implicit, attributes in the relation profile associated with each node $n \in$ N.

- *Assignments.* For each subject $s \in \mathcal{S}$ and each node $n \in$ N, there is a binary variable $\lambda_{s,n}$ that is equal to 1 if node n is assigned to subject s for evaluation; it is 0, otherwise.
- *Profiles.* For each attribute $a \in A$ and each node $n \in$ N, there are four binary variables $vp_{a,n}$, $ve_{a,n}$, $ip_{a,n}$, and $ie_{a,n}$. Variable $vp_{a,n}$ ($ve_{a,n}$, $ip_{a,n}$, $ie_{a,n}$, resp.) is equal to 1 if a is a visible plaintext (visible encrypted, implicit plaintext, implicit encrypted, resp.) attribute in the profile of node n; it is 0, otherwise.

Input variables. They model the input of the problem, namely: the query plan tree, the candidates for each node (step 1), the attributes involved in the evaluation of each node (needed to guarantee uniform encrypted/plaintext representation of attributes that need to be compared and to set the implicit component in the node profile), the authorizations (needed to determine possible encryption that has to be enforced on some attributes to make candidates authorized for the nodes they have been assigned), and the profiles of nodes over minimum required views (needed to guarantee the correctness of node profiles in terms of the com-

pleteness of the set of visible and implicit attributes and to ensure the operation feasibility through the plaintext representation in the profiles of the attributes needed for the operation's evaluation). For readability, in the reminder of this section, we denote minimum required views with MRVs and, when referring to the profile resulting from executing the operation of a node on the MRVs over its children (i.e., with reference to our running example, to the profiles in Figure 7), we will omit "over its children".

- *Tree structure.* For each pair of nodes $\bar{n},n\in\mathtt{N}$, there is a variable $\pi_{\bar{n},n}$ that is set to 1 if $\bar{n}$ is the parent of n in the tree; it is 0, otherwise.
- *Candidates.* For each subject $s \in \mathcal{S}$ and each node $n \in \mathtt{N}$, there is a variable $c_{s,n}$ that is set to 1 if subject s is a candidate for n; it is 0, otherwise.
- *Attributes involved in operations.* For each attribute $a \in A$ and each node $n \in \mathtt{N}$, there is a variable $\iota_{a,n}$ that is set to 1 if the operation represented by node n inserts a into the implicit component of the relation profile of n; it is 0, otherwise. Furthermore, for each node $n \in \mathtt{N}$ and each pair of attributes a and $\bar{a}$, there is a variable $\hat{eq}_{a,\bar{a},n}$ that is set to 1 if the operation represented by node n defines an equivalence relationship between a and $\bar{a}$; it is 0, otherwise.
- *Authorizations.* For each subject $s \in \mathcal{S}$ and each attribute $a \in A$, there are two variables, $auth\_p_{s,a}$ and $auth\_e_{s,a}$. Variable $auth\_p_{s,a}$ ($auth\_e_{s,a}$, resp.) is set to 1 if subject s is authorized to access attribute a in plaintext (encrypted, resp.) form; it is 0, otherwise.
- *Profiles over minimum required views.* For each attribute $a \in A$ and each node $n \in \mathtt{N}$, there are four variables $\hat{vp}_{a,n}$, $\hat{ve}_{a,n}$, $\hat{ip}_{a,n}$, and $\hat{ie}_{a,n}$. Variable $\hat{vp}_{a,n}$ ($\hat{ve}_{a,n}$, $\hat{ip}_{a,n}$, $\hat{ie}_{a,n}$, resp.) is set to 1 if a belongs to the visible plaintext (visible encrypted, implicit plaintext, implicit encrypted, resp.) component of the profile of n on the MRVs; it is 0, otherwise. Note that for each leaf node n (representing a base relation), these four variables correspond to the minimun required view over n for the execution of the parent node $\bar{n}$.

6.2 Constraints

Constraints restrict the combination of values for the output variables described in Section 6.1 to guarantee that the solution computed for our binary programming problem represents a minimum cost assignment for the query plan given in input. Such constraints are formulated as follows.

Authorized assignment. This set of constraints guarantees that the values of variables $\lambda_{s,n}$ represent an assignment function that is compliant with the candidate assignment function and the authorizations.

- *Each node in the query plan is assigned to exactly one subject.*

  $$\forall n\in\mathtt{N}\colon \textstyle\sum_{s\in\mathcal{S}} \lambda_{s,n}=1 \tag{1}$$

  Intuitively, for each node n, the constraint sums $\lambda_{s,n}$ over all the subjects in $\mathcal{S}$. If the sum is equal to 1, then there exists only one subject s such that $\lambda_{s,n}$ is equal to 1, meaning that there exists only one subject to which n has been assigned.
- *Assignees are candidates for nodes they have been assigned.*

  $$\forall n\in\mathtt{N}\colon \textstyle\sum_{s\in\mathcal{S}} \lambda_{s,n}\cdot c_{s,n}=1 \tag{2}$$

  The product $\lambda_{s,n}\cdot c_{s,n}$ is equal to 1 only if the operation at node n is assigned to subject s ($\lambda_{s,n}=1$) and s is a candidate for n ($c_{s,n}=1$).
- *The extended query plan ensures assignees are authorized for the nodes they have been assigned.* Assigning a node to one of its candidates guarantees that the assignee has sufficient authorization for executing the operation (Theorem 3), which however can be provided extending the query plan with encryption to cover attributes the assignee cannot access plaintext. The profile of the nodes in the extended query plan returned should then be compliant with authorizations. In other words, the assignee of a node must be authorized for all the attributes in the profile of the node and visibility (plaintext or encrypted) should be compliant with authorizations.

  $\forall a\in A, \forall n\in\mathtt{N}, \forall s\in\mathcal{S}$:

  $$vp_{a,n}\cdot\lambda_{s,n}\leq auth\_p_{s,a} \tag{3}$$
  $$ip_{a,n}\cdot\lambda_{s,n}\leq auth\_p_{s,a} \tag{4}$$
  $$ve_{a,n}\cdot\lambda_{s,n}\leq auth\_p_{s,a}+auth\_e_{s,a} \tag{5}$$
  $$ie_{a,n}\cdot\lambda_{s,n}\leq auth\_p_{s,a}+auth\_e_{s,a} \tag{6}$$

  The products $vp_{a,n}\cdot\lambda_{s,n}$ and $ip_{a,n}\cdot\lambda_{s,n}$ ($ve_{a,n}\cdot\lambda_{s,n}$, $ie_{a,n}\cdot\lambda_{s,n}$, resp.) are equal to 1 if the operation at node n is assigned to s and attribute a is visible plaintext or implicit plaintext (visible encrypted or implicit encrypted, resp.) in the profile of the relation resulting from the evaluation of n. If the product is equal to 1, the constraint is satisfied only if also $auth\_p_{s,a}=1$ ($auth\_p_{s,a}+auth\_e_{s,a}=1$, resp.) and hence s is authorized to access a in plaintext (in plaintext or encrypted, resp.). If the product is equal to 0 the constraint is always satisfied, independently from authorizations.

Integrity of the profiles. This set of constraints guarantees integrity of the profiles, meaning that they capture the informative content of base and derived relations as discussed in Section 3.2.

– *Attribute representation in schema.* An attribute cannot appear in each node more than once (i.e., in both the plaintext and encrypted component).

$$\forall a{\in}A, \forall n{\in}\mathtt{N}\colon\ vp_{a,n}+ve_{a,n}{\leq}1 \tag{7}$$

If an attribute is represented both in plaintext and encrypted in a node, both $vp_{a,n}$ and $ve_{a,n}$ are equal to 1, hence their sum is 2 violating the constraint.

– *Attribute representation in implicit component.* An attribute cannot appear in the implicit component more than once (i.e., in both the implicit plaintext and implicit encrypted component).

$$\forall a{\in}A, \forall n{\in}\mathtt{N}\colon\ ip_{a,n}+ie_{a,n}{\leq}1 \tag{8}$$

If an attribute is represented both in plaintext and encrypted in a node, both $ip_{a,n}$ and $ie_{a,n}$ are equal to 1, hence their sum is 2, violating the constraint.

– *Base relations have all their attributes in plaintext and have no implicit attributes.* The profile of a base relation R includes all and only the attributes in the relation schema in plaintext. The implicit component is empty and no attribute is encrypted. This specific format of the relation profile translates in the following constraints that need to be satisfied by the relation profiles associated with nodes representing base relations.

$$\forall n{\in}\{n{\in}\mathtt{N}{:}\pi_{n,\bar{n}}{=}0, \forall \bar{n}{\in}\mathtt{N}\}\colon$$

$$vp_{a,n}{=}1, \forall a{\in}R \tag{9}$$

$$vp_{a,n}+ve_{a,n}{=}0, \forall a{\notin}R \tag{10}$$

$$ip_{a,n}+ie_{a,n}{=}0, \forall a{\in}A \tag{11}$$

Here, R is the relation represented by node n, and set $\{n{\in}\mathtt{N}{:}\pi_{n,\bar{n}}{=}0, \forall \bar{n}{\in}\mathtt{N}\}$ contains all leaf nodes in $\mathtt{N}$. Constraints above require then that: each attribute a in the schema of a base relation (i.e., $a{\in}R$) is visible plaintext in its profile; for each attribute a that does not appear in the visible component of R, variables $vp_{a,n}$ and $ve_{a,n}$ are set to 0 (i.e., modeling the fact that the attribute does not belong to R); and for all attributes $a \in A$, variables $ip_{a,n}$ and $ie_{a,n}$ are both set to 0 (i.e., the implicit component in the profile of base relations is empty).

Support for query evaluation. This set of constraints models the requirements that impose the plaintext/encrypted representation of attributes in the profile of nodes to support the execution of the nodes' operations. In fact, for the execution of an operation, some attributes might be required to be represented in plaintext (e.g., for the evaluation of a selection condition) or be represented in the same form (e.g., both plaintext or encrypted for comparing their values).

– *Operation feasibility.* For each node, attributes that are requested to be in plaintext for supporting the execution of the operation at the node appear plaintext in the node, as dictated by the profile of the node on the MRVs.

$$\forall a{\in}A, \forall n{\in}\mathtt{N}\colon\ vp_{a,n}{\geq}\hat{vp}_{a,n} \tag{12}$$

If an attribute a is plaintext in the profile of node n on the MRVs, $\hat{vp}_{a,n}$ is equal to 1. In this case, the constraint imposes that also the value of $vp_{a,n}$ is equal to 1, that is, the attribute must be in plaintext also in the node profile of the computed solution.

– *Comparison feasibility.* In each node, attributes that need to be used together (i.e., compared or input to an udf) must appear in the same (either plaintext or encrypted) form.

$$\forall a, \bar{a}{\in}A, \forall n{\in}\mathtt{N}\colon \tag{13}$$
$$(\hat{eq}_{a,\bar{a},n}\cdot vp_{a,n}\cdot vp_{\bar{a},n})+(\hat{eq}_{a,\bar{a},n}\cdot ve_{a,n}\cdot ve_{\bar{a},n}){=}\hat{eq}_{a,\bar{a},n}$$

For each node n, the constraint is specified for each pair of attributes a and $\bar{a}$ that are used together in the node, expressed by input variable $\hat{eq}_{a,\bar{a},n}$ equal to 1. The product $\hat{eq}_{a,\bar{a},n}\cdot vp_{a,n}\cdot vp_{\bar{a},n}$ is equal to 1 iff a and $\bar{a}$ are compared and are both visible plaintext in the profile of n. Analogously, the product $\hat{eq}_{a,\bar{a},n}\cdot ve_{a,n}\cdot ve_{\bar{a},n}$ is equal to 1 iff a and $\bar{a}$ are used together and are both visible encrypted. Note that the sum of such products is constrained to be equal to $\hat{eq}_{a,\bar{a},n}$, meaning that if the attributes are used together (i.e., $\hat{eq}_{a,\bar{a},n}{=}1$), then they are either both plaintext or both encrypted.

Profile correctness. This set of constraints specifies correctness criteria on the profile of nodes associated with the extended query plan of the solution. It captures correctness of each node's profile, which must take into account all attributes of its operands (as expressed in the MRVs), and the correct computation of implicit information, which must take into account implicit attributes carried by the operands as well as new implicit information originated by the node's operation.

– *Compliance with MRVs.* For each node, all attributes visible (implicit, resp.) in the profile of the node on the MRVs must be visible (implicit, resp.) in the node. Intuitively, such compliance dictates

that no attribute can be lost or added in the node profile.

$\forall a \in A, \forall n \in \mathtt{N}$:

$$vp_{a,n} + ve_{a,n} = \hat{vp}_{a,n} + \hat{ve}_{a,n} \quad (14)$$

$$ip_{a,n} + ie_{a,n} = \hat{ip}_{a,n} + \hat{ie}_{a,n} \quad (15)$$

Note that, for each node, only one of the variables at any side of the equality can be equal to 1, as an attribute cannot appear both encrypted and plaintext in the schema of a node (or in the implicit component). This mutual exclusion is guaranteed for the attributes in the profile of the node on the MRVs as they are provided as input, and for the attributes in the solution's profile by the attribute representation constraints (Constraints 7 and 8). The equalities above then require attributes in the profile of a node to be all and only the attributes in the profile of the node on the MRVs. Note that the constraints only impose that attributes do not appear or disappear from the schema (or the implicit component), but they do not impose that attributes must be in the same form (i.e., encrypted or plaintext) as this depends on the specific assignment of the solution (which might need - or not need - to inject encryption to cover attributes that the assignee cannot access in plaintext).

– *Correctness of the implicit component.* For each node, the (plaintext/encrypted) implicit component in the profile must include all the (plaintext/encrypted) implicit attributes carried by the children (i.e., in the implicit components of the children) as well as all (plaintext/encrypted) attributes involved in a selection or group by operation in the node.

$\forall a \in A, \forall n, \bar{n} \in \mathtt{N}$:

$$ip_{a,n} \geq ip_{a,\bar{n}} \cdot \pi_{n,\bar{n}} \quad (16)$$

$$ie_{a,n} \geq ie_{a,\bar{n}} \cdot \pi_{n,\bar{n}} \quad (17)$$

$\forall a \in A, \forall n \in \mathtt{N}$:

$$ip_{a,n} \geq \iota_{a,n} \cdot vp_{a,n} \quad (18)$$

$$ie_{a,n} \geq \iota_{a,n} \cdot ve_{a,n} \quad (19)$$

The first two constraints require $ip_{a,n}$ ($ie_{a,n}$, resp.) to be 1 whenever $ip_{a,\bar{n}}$ ($ie_{a,\bar{n}}$, resp.) is 1 in at least a child of n; enforcing propagation in n's profile of the implicit attributes of its children. The latter two constraints require $ip_{a,n}$ ($ie_{a,n}$, resp.) to be equal to 1 for any attribute a such that $vp_{a,n}$=1 ($ve_{a,n}$=1, resp.), meaning that the attribute is in the schema of node n, and the attribute has been involved in a selection or group by operation in n (i.e., $\iota_{a,n}$ is 1). These constraints then impose n's implicit component to include the attributes that affected n's computation (i.e., attributes that leave a trace in the result).

$cost_s$:	CPU usage cost of subject s
tr_cost_s:	outbound data transfer cost of subject s
$eval\_eff_n$:	computational effort for the execution of n
$eeff_a$:	computational effort for encrypting attribute a
$deff_a$:	computational effort for decrypting attribute a
$size_a$:	size of attribute a
$esize_a$:	size of the encrypted version of attribute a
$ocard_n$:	cardinality of the relation resulting from n

Fig. 10 Cost parameters

6.3 Objective function

The objective function of the binary programming problem models the economic cost of the evaluation of the extended authorized query plan $\mathtt{T'(N)}$ for λ', with $\lambda'(n)=\lambda_{s,n}$, $\forall n \in \mathtt{N}$. The cost is computed as the sum of three components: *i)* the computational cost OP_EXEC of evaluating each node in the query plan; *ii)* the encryption/decryption cost ENC_DEC of enforcing encryption and decryption operations; and *iii)* the transmission cost TRANSF of data among subjects. Formally, the objective function is defined as:

$$\textbf{min}(\mathsf{OP\_EXEC}+\mathsf{ENC\_DEC}+\mathsf{TRANSF})$$

We now describe each of the three cost components. Figure 10 summarizes the cost parameters that will be used in such cost components.

Operation execution. The cost of executing the operations in the query plan is the sum of the costs of executing the different nodes' operations at the subject to which they have been assigned. For each node $n \in \mathtt{N}$, such a cost depends on the CPU usage cost $cost_s$ of the subject s to which the node has been assigned (i.e., for which $\lambda_{s,n}$ is equal to 1), multiplied by the computational effort $eval\_eff_n$ required for the execution, which depends on the operation to be executed and on the size of the input to be processed. The computational cost of the query is then computed as:

$$\mathsf{OP\_EXEC}=\sum_{n \in \mathtt{N}, s \in \mathcal{S}}(\lambda_{s,n} \cdot cost_s \cdot eval\_eff_n)$$

Encryption/decryption. Encryption and decryption operations must be executed whenever the representation of a visible attribute changes from a node n to its parent $\bar{n}$ (from visible plaintext to visible encrypted or vice versa), as dictated by the extended query plan

determined from the solution of the minimization problem. A change from plaintext to encrypted requires the execution of an encryption operation by the child node(s)' assignee(s) before data are transmitted to the parent node's assignee. A change from encrypted to plaintext requires the execution of a decryption operation by the parent node's assignee upon reception of data from the child(ren) assignee(s). The costs considered for the operation are therefore the computational cost ($cost_s$) of the subject performing the operation (i.e., subject s for which $\lambda_{s,n}$=1 for encryption and $\lambda_{s,\bar{n}}$=1 for decryption), the computational effort of such operation ($eeff_a$ and $deff_a$, resp.), which depends on the encryption scheme used for the attribute involved, and the size of the data to be encrypted (decrypted, resp.). This latter is computed, for each attribute involved in the operation, as the size ($size_a$ for encryption and $esize_a$ for decryption) of the attribute multiplied for the cardinality of the operand ($ocard_n$), where the distinction between $size_a$ and $esize_a$ takes into account the fact that encryption can increase the size of the attribute. The cost of the execution of encryption and decryption operations is then computed as:

$$\begin{aligned}\mathsf{ENC\_DEC} = \sum_{n\in\mathtt{N},\bar{n}\in\mathtt{N},s\in\mathcal{S},a\in A}(&\lambda_{s,n}\cdot\pi_{\bar{n},n}\cdot vp_{a,n}\cdot ve_{a,\bar{n}}\cdot\\ &cost_s\cdot eeff_a\cdot size_a\cdot ocard_n) \;+\; (\lambda_{s,\bar{n}}\cdot\pi_{\bar{n},n}\cdot\\ &vp_{a,\bar{n}}\cdot ve_{a,n}\cdot cost_s\cdot deff_a\cdot esize_a\cdot ocard_n)\end{aligned}$$

The first part of this formula computes the cost of encryption operations, and the second one computes the cost of decryption operations. Note that, product $\pi_{\bar{n},n}\cdot vp_{a,n}\cdot ve_{a,\bar{n}}$ ($\pi_{\bar{n},n}\cdot vp_{a,\bar{n}}\cdot ve_{a,n}$, resp.) is equal to 1 only if a appears plaintext in n and encrypted in the parent node $\bar{n}$ (or vice versa).

Data transfer. The cost of data transfer refers to the cost involved for transferring data from one subject to another that occurs whenever a node n in the query plan and its parent $\bar{n}$ are assigned to different subjects. Since inbound data transfer is usually free, we consider only the outbound data transfer cost, that is, the cost for the assignee s of the child node (such that $\lambda_{s,n}$=1) to send out its results. The cost of such operation is the transfer cost tr_cost_s of such subject s multiplied by the amount of data to be transferred, which is in turn given by the cardinality of the relation multiplied by the size of the attributes. The latter is the plaintext size ($size_a$) if the attribute is transmitted plaintext, that is, it appears plaintext in both n and $\bar{n}$ profiles (i.e., $vp_{a,n}\cdot vp_{a,\bar{n}}$ is 1); it is the encrypted size ($esize_a$) if the attribute is transmitted encrypted, that is, it either appears encrypted in n (i.e., $ve_{a,n}$ is equal to 1) or it appears plaintext in n but encrypted in $\bar{n}$ (i.e., $vp_{a,n}\cdot ve_{a,\bar{n}}$ is equal to 1), meaning that the attribute has to be encrypted before transmission. The cost of transferring data is then computed as:

$$\begin{aligned}\mathsf{TRANSF} = \sum_{n\in\mathtt{N},\bar{n}\in\mathtt{N},s\in\mathcal{S},a\in A}\lambda_{s,n}\cdot(1-\lambda_{s,\bar{n}})\cdot\pi_{\bar{n},n}\cdot\\ tr\_cost_s\cdot ocard_n\cdot((size_a\cdot vp_{a,n}\cdot vp_{a,\bar{n}})+\\ (esize_a\cdot(ve_{a,n}+(vp_{a,n}\cdot ve_{a,\bar{n}}))))\end{aligned}$$

Here, product $\lambda_{s,n}\cdot(1-\lambda_{s,\bar{n}})\cdot\pi_{\bar{n},n}$ is equal to 1 whenever nodes n and $\bar{n}$, with $\bar{n}$ parent of n (i.e., for which $\pi_{\bar{n},n}$ is equal to 1) are such that the assignee s of n (i.e., for which $\lambda_{s,n}$ is equal to 1) is not the assignee of $\bar{n}$ (i.e., $\lambda_{s,\bar{n}}$ is equal to 0).

Note that the cost of data transfer must also include the cost for sending the query result to the user (which might not be the assignee of the root of the query tree plan). We model such a cost by adding a node at the root of the query tree plan. Such a node does not correspond to any operation and has the same profile as the original root, except for the fact that all the visible attributes appear in plaintext since the assignee is forced to be the user who submitted the query. This extra node permits to keep into consideration the cost of transferring the query result to the user as well as the cost of the decryption operations performed by the user on the encrypted attributes in the query result.

Figure 11 summarizes the formulation of our binary programming problem for computing a minimum cost assignment. The solution gives a value to the output variables in Figure 9, modeling assignment of nodes' operations to subjects (value of variables $\lambda_{s,n}$), and the attributes that appear in the visible and implicit component of the profiles along with their plaintext/encrypted representation (value of variables $vp_{a,n}$, $ve_{a,n}$, $ip_{a,n}$, and $ie_{a,n}$). This induces the natural injection of encryption and decryption in the query plan, resulting in an extended authorized query plan, thus solving Problem 1.

7 Computing and distributing assignments

In this section, we discuss some aspects related to encryption and authorization enforcement in the actual execution of the extended query plan.

Key distribution. Query operation assignment entails, besides assigning operations to candidates, also establishing and distributing keys for attributes that need to be encrypted/decrypted in the query plan execution. The only constraint on key establishment is that attributes involved in some conditions comparing them in encrypted form need to be encrypted with the same key. To ensure this, we simply require attributes appearing together in an equivalence set to be encrypted

min

OP_EXEC $\sum_{n\in\mathtt{N},s\in\mathcal{S}} (\lambda_{s,n}\cdot cost_s\cdot eval\_eff_n)+$

ENC_DEC $\sum_{n\in\mathtt{N},\bar{n}\in\mathtt{N},s\in\mathcal{S},a\in A} (\lambda_{s,n}\cdot\pi_{\bar{n},n}\cdot vp_{a,n}\cdot ve_{a,\bar{n}}\cdot cost_s\cdot eeff_a\cdot size_a\cdot ocard_n)+$
$(\lambda_{s,\bar{n}}\cdot\pi_{\bar{n},n}\cdot vp_{a,\bar{n}}\cdot ve_{a,n}\cdot cost_s\cdot deff_a\cdot esize_a\cdot ocard_n)+$

TRANSF $\sum_{n\in\mathtt{N},\bar{n}\in\mathtt{N},s\in\mathcal{S},a\in A} (\lambda_{s,n}\cdot(1-\lambda_{s,\bar{n}})\cdot\pi_{\bar{n},n}\cdot tr\_cost_s\cdot ocard_n\cdot((size_a\cdot vp_{a,n}\cdot vp_{a,\bar{n}})+(esize_a\cdot(ve_{a,n}+(vp_{a,n}\cdot ve_{a,\bar{n}}))))$

s.t.

(1) $\sum_{s\in\mathcal{S}}\lambda_{s,n}=1,\quad \forall n\in\mathtt{N}$

(2) $\sum_{s\in\mathcal{S}}\lambda_{s,n}\cdot c_{s,n}=1,\quad \forall n\in\mathtt{N}$

(3) $vp_{a,n}\cdot\lambda_{s,n}\leq auth\_p_{s,a},\quad \forall a\in A,\forall n\in\mathtt{N},\forall s\in\mathcal{S}$

(4) $ip_{a,n}\cdot\lambda_{s,n}\leq auth\_p_{s,a},\quad \forall a\in A,\forall n\in\mathtt{N},\forall s\in\mathcal{S}$

(5) $ve_{a,n}\cdot\lambda_{s,n}\leq auth\_p_{s,a}+auth\_e_{s,a},\forall a\in A,\quad \forall n\in\mathtt{N},\forall s\in\mathcal{S}$

(6) $ie_{a,n}\cdot\lambda_{s,n}\leq auth\_p_{s,a}+auth\_e_{s,a},\forall a\in A,\quad \forall n\in\mathtt{N},\forall s\in\mathcal{S}$

(7) $vp_{a,n}+ve_{a,n}\leq 1,\quad \forall a\in A,\forall n\in\mathtt{N}$

(8) $ip_{a,n}+ie_{a,n}\leq 1,\quad \forall a\in A,\forall n\in\mathtt{N}$

(9) $vp_{a,n}=1,\quad \forall a\in R,\forall n\in\{n\in\mathtt{N}:\pi_{n,\bar{n}}=0,\forall\bar{n}\in\mathtt{N}\}$

(10) $vp_{a,n}+ve_{a,n}=0,\quad \forall a\in A\setminus(R^{vp}\cup R^{ve}),\forall n\in\{n\in\mathtt{N}:\pi_{n,\bar{n}}=0,\forall\bar{n}\in\mathtt{N}\}$

(11) $ip_{a,n}+ie_{a,n}=0,\quad \forall a\in A,\forall n\in\{n\in\mathtt{N}:\pi_{n,\bar{n}}=0,\forall\bar{n}\in\mathtt{N}\}$

(12) $vp_{a,n}\geq\hat{vp}_{a,n},\quad \forall a\in A,\forall n\in\mathtt{N}$

(13) $(\hat{eq}_{a,\bar{a},n}\cdot vp_{a,n}\cdot vp_{\bar{a},n})+(\hat{eq}_{a,\bar{a},n}\cdot ve_{a,n}\cdot ve_{\bar{a},n})=\hat{eq}_{a,\bar{a},n},\quad \forall a,\bar{a}\in A,\forall n\in\mathtt{N}$

(14) $vp_{a,n}+ve_{a,n}=\hat{vp}_{a,n}+\hat{ve}_{a,n},\quad \forall a\in A,\forall n\in\mathtt{N}$

(15) $ip_{a,n}+ie_{a,n}=\hat{ip}_{a,n}+\hat{ie}_{a,n},\quad \forall a\in A,\forall n\in\mathtt{N}$

(16) $ip_{a,n}\geq ip_{a,\bar{n}}\cdot\pi_{n,\bar{n}},\quad \forall a\in A,\forall n,\bar{n}\in\mathtt{N}$

(17) $ie_{a,n}\geq ie_{a,\bar{n}}\cdot\pi_{n,\bar{n}},\quad \forall a\in A,\forall n,\bar{n}\in\mathtt{N}$

(18) $ip_{a,n}\geq\iota_{a,n}\cdot vp_{a,n},\quad \forall a\in A,\forall n\in\mathtt{N}$

(19) $ie_{a,n}\geq\iota_{a,n}\cdot ve_{a,n},\quad \forall a\in A,\forall n\in\mathtt{N}$

Fig. 11 Binary programming for computing a minimum cost assignment

with the same key (even if they are encrypted after they have been compared, using the same key would not provide any leakage as they are equivalent). As per Theorem 1, it is sufficient to look at the equivalence sets in the profile of the root to determine which attributes should be encrypted with the same key. For instance, consider the extended authorized query plans in Figure 8. Attributes S and C must be encrypted with the same key since they belong to the equivalence set of the profile of the root node. We then define the keys to be established for a query plan execution as follows.

Definition 10 (Query Plan Keys) Let T(N) be an extended authorized query plan, n_T be its root, and A_k be the set of attributes involved in encryption operations. Let $\mathcal{A}=\{\{A_k\cap A_j\}|A_j\in R^{\widetilde{\leq}}_T\}\cup\{\{a\}|a\in A_k,\nexists A_j\in R^{\widetilde{\leq}}_T,a\in A_j\}$. The set $\mathcal{K}_T$ of keys for T is $\mathcal{K}_T=\{k_A|A\in\mathcal{A}\}$, with k_A an encryption key.

In the definition, the set of sets $\mathcal{A}$ clusters attributes to be encrypted based on the equivalence sets in the root profile (attributes appearing together in an equivalence set belong to the same set in $\mathcal{A}$, while attributes not belonging to any equivalence set appear as singletons). The key associated with an attribute (or set thereof) will be distributed only to the subjects in charge for its (their) encryption, and possible decryption. Since such subjects are authorized for the encryption/decryption operation (i.e., they are authorized for plaintext visibility of the attributes to be encrypted/decrypted in the operand relation), key distribution obeys authorizations. For instance, for the query plan in Figure 8(a), $\mathcal{A}$={SC,P}, resulting in k_{SC} distributed to $\mathbb{H}$ and $\mathbb{I}$, and k_{P} distributed to $\mathbb{I}$ and $\mathbb{Y}$. For the query plan in Figure 8(b), $\mathcal{A}$={D,P}, resulting in k_{D} distributed to $\mathbb{H}$, and k_{P} distributed to $\mathbb{I}$ and $\mathbb{Y}$.

Encryption algorithm. As stated in Section 2, our authorization model does not distinguish among different encryption schemes. The query optimizer should however choose the scheme (e.g., deterministic or randomized encryption) depending on the operation that has to be executed on the resulting encrypted values [13,29]. As a matter of fact, the ability to operate on encrypted data (Figure 12) comes with possible exposure to inference as well as with a cost. For instance, direct encryption can be exposed to frequency attacks, while order preserving encryption leaks order relation-

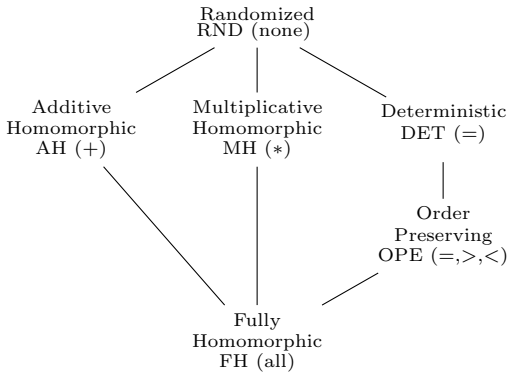


Fig. 12 Functionality of encryption schemes

S	Receives ($\text{req}_\mathbb{S}$)	Performs ($q_\mathbb{S}$)
$\mathbb{Y}$	$[[q_\mathbb{Y},(\text{P},k_\text{P})]_{\text{pri}_\mathbb{U}}]_{\text{pub}_\mathbb{Y}}$	SELECT T,`decrypt`(P^k,k_P) AS P FROM $[\![\text{req}_\mathbb{X}]\!]$ WHERE P >100
$\mathbb{X}$	$[[q_\mathbb{X},\text{-}]_{\text{pri}_\mathbb{U}}]_{\text{pub}_\mathbb{X}}$	SELECT T,avg(P^k) AS P^k FROM $[\![\text{req}_\mathbb{H}]\!]$ JOIN $[\![\text{req}_\mathbb{I}]\!]$ ON S^k=C^k GROUP BY T
$\mathbb{H}$	$[[q_\mathbb{H},(\text{S},k_\text{SC})]_{\text{pri}_\mathbb{U}}]_{\text{pub}_\mathbb{H}}$	SELECT `encrypt`(S,k_SC),D,T FROM Hosp WHERE D='stroke'
$\mathbb{I}$	$[[q_\mathbb{I},(\text{C},k_\text{SC})(\text{P},k_\text{P})]_{\text{pri}_\mathbb{U}}]_{\text{pub}_\mathbb{I}}$	SELECT `encrypt`(C,k_SC),`encrypt`(P,k_P) FROM Ins

Fig. 13 Query dispatch for the plan in Figure 8(a)

ships of data. We propose to adopt, for each attribute, the scheme providing the highest protection, while supporting the operations to be executed on the attribute's encrypted values. For instance, if for an attribute no operation needs to be executed on encrypted values, randomized encryption is used, while if equality conditions need to be evaluated, deterministic encryption is used. Similarly, additive homomorphic (e.g., Paillier) or multiplicative homomorphic (e.g., ElGamal) schemes are applied when only sums or products need to be executed over attribute values. Each attribute can be encrypted with a different encryption scheme and with a different key, the only constraint is that attributes that are involved together in some operations (i.e., attributes that belong to the same set in the equivalence set of the root's profile) need to be encrypted with the same key to enable the execution of the operations.

Query dispatch. The query dispatch operates according to classical approaches, with the only difference that subjects may be communicated keys and they may need to execute, in addition to operations requested by query computation, also encryption and decryption operations. We assume each subject S involved in a query plan to have a private (pri_S), public (pub_S) key pair. The communication to each subject will be signed with the private key of the user and encrypted with the subject's public key. Having a sub-query signed allows the recipient to verify its authenticity and integrity. Encrypting a sub-query with the public key of the recipient supports confidentiality of the communication. Note, however, that the correctness of our approach does not depend on the simple protection of the communication. As a matter of fact, the definition of profiles does not make any assumption on the confidentiality of the query, which could potentially be known (of course with conditions operating on encrypted values when demanded by encryption operations in the plan). Figure 13 illustrates the query dispatch for the plan in Figure 8(a). In the figure, term $\text{req}_\mathbb{S}$=$[[q_S,(A_1,k_1),\ldots,(A_n,k_n)]_{\text{pri}_\mathbb{U}}]_{\text{pub}_S}$ represents the request (signed with $\text{pri}_\mathbb{U}$ and encrypted with pub_S) sent to subject S, where q_S is the sub-query and $(A_1,k_1),\ldots,(A_n,k_n)$ is the list of attributes that subject S must encrypt/decrypt with the corresponding key. The plan starts with the request from $\mathbb{U}$ to $\mathbb{Y}$ ($\text{req}_\mathbb{Y}$), which will call the sub-query at $\mathbb{X}$ ($\text{req}_\mathbb{X}$), which in turn will call the sub-queries at $\mathbb{H}$ ($\text{req}_\mathbb{H}$) and $\mathbb{I}$ ($\text{req}_\mathbb{I}$).

Authorization enforcement. Our approach relies on the correct enforcement of authorizations throughout the query plan. Since the definition of the query plan is outside the control of the involved data authorities, the query optimizer has to be trusted for such an enforcement. Each data authority will perform a control at its side, before releasing the data to a third party, to check that the user is authorized for the released data. In fact, a user requesting query execution must be authorized to access all data that are input to the query, which correspond to the base relations. The user is then trusted to involve other authorized subjects. With respect to the authorization enforcement, in the description of our approach, for simplicity, we have assumed the control of the authorizations holding for a given subject simply as a check against the set $\mathcal{P}_S$ ($\mathcal{E}_S$, resp.) summarizing the attributes that subject S is authorized to access plaintext (in encrypted form, resp.). While the realization of such a control directly against a global repository storing $\mathcal{P}_S$ and $\mathcal{E}_S$, for all subjects, can be possible, in real applications we can expect authorizations over the different relations to be stored in a distributed manner, like the relations are, and remain under the control of the respective data authorities. This distributed storage and management of authorizations is completely in line with our approach. As a matter of fact, a major advantage of the consideration of authorizations holding only on specific relations (no cross-relations/cross-authority authorizations) is that it simplifies authorizations specification and management and makes our solution completely independent from the approach adopted for storing and managing autho-

rizations. For instance, a data authority can: *i)* publish its access control policy (which would then result publicly visible), or *ii)* respond to explicit authorization requests. The first approach can facilitate access to the policy, but entails its complete exposure. The second approach has instead the advantage of maintaining the whole policy confidential, providing only the responses to individual authorization checks. Our proposal is independent of the specific approach adopted and can work with both of them.

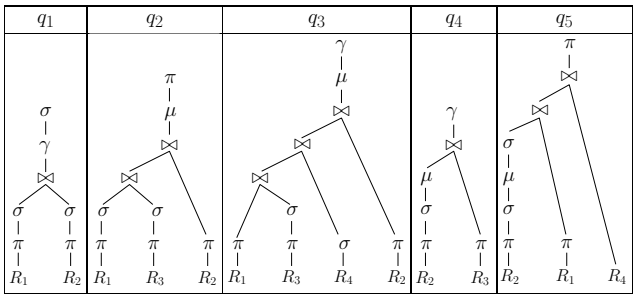


Fig. 14 Query plans for the use-case

8 Experimental results

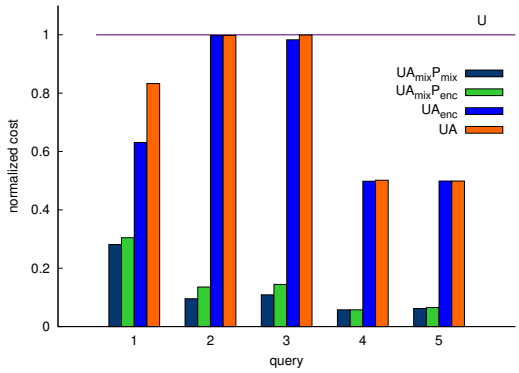
Our authorization model, supporting and enforcing encrypted visibility to external authorities and providers, enables the delegation of intensive computation to external parties in a way that produces the greatest advantage in the query execution. Such ability to delegate computation to providers with the lowest cost among those trusted to access (in plaintext or encrypted form) the involved data can bring considerable advantages since even small reductions in price lead to a reduction in the economic costs associated with the execution of queries. To evaluate the economic benefits of our approach in distributed query execution, which enables to fully enjoy the economic benefit of the open cloud market, we realized a tool implementing the two steps illustrated in Section 6 for computing a minimum cost assignment for a query plan and performed a series of experiments. We implemented the first step, computing the candidate assignment function Λ, in Java and the second step, computing the minimum cost assignment, using LINGO[2] for solving our binary programming problem. Our tool receives in input a relational schema, a query plan, the price lists of each subject (i.e., user, data authorities, and cloud providers) for cpu usage and data transfer, and the authorizations. It provides as output a minimum cost assignment of operations in the query plan to subjects, and the corresponding extended authorized query plan, introducing the encryption and decryption operations needed to make the computed assignment authorized and enable the evaluation of operations.

Queries. Aiming at considering a scenario with queries explicitly using udf functionalities (which are not used in existing benchmarks), we considered queries representative of a use-case provided by a large manufacturing company that applies data analysis to extract information from production data combined with customers data and data provided by external agencies. These analyses typically require the execution of udfs. The queries operate on four relations, distributed among three data authorities. The query plans differ in: *1)* the number of relations involved (ranging from 2 to 4), and *2)* the position in the query tree plan where the udf operates (i.e., close to a base relation, and hence operating on the data owned by a single authority, or up in the query plan, and hence operating on the result of computations combining data of different authorities). Figure 14 provides a high-level representation (omitting attributes and operation parameters) of the query plans. In particular, query q_1 involves two relations and has no udf; q_2 involves three relations and the udf operates close to the root; q_3 involves four relations and the udf operates close to the root; q_4 involves two relations and the udf operates close to a base relation; and q_5 involves three relations and the udf operates close to a base relation. We estimated the size of processed data, the increase in size that may derive from the application of encryption, and the computational costs of relational operators and of encryption/decryption operations based on the estimates produced by query optimizers and on common benchmarks.

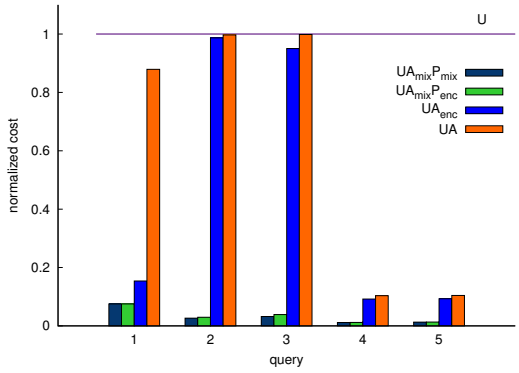
Authorization scenarios. We considered different authorization scenarios with increasing visibility over data by authorities and external providers.

- **UA**: authorizations allow the user to access all data and data authorities to access their own base relations.
- $\mathbf{UA_{enc}}$: enriches the previous scenario with authorizations allowing each data authority to access, in encrypted form, all the attributes in the base relations owned by the other data authorities.
- $\mathbf{UA_{mix}P_{enc}}$: enriches the previous scenario with authorizations that allow authorities to access in plaintext some of the attributes in the base relations owned by the other authorities, and cloud providers to access, in encrypted form, all the attributes of all the base relations.
- $\mathbf{UA_{mix}P_{mix}}$: enriches the previous scenario with authorizations allowing cloud providers to access in

[2] LINGO https://www.lindo.com/index.php/products/lingo-and-optimization-modeling



(a) 2-20 configuration



(b) 10-100 configuration

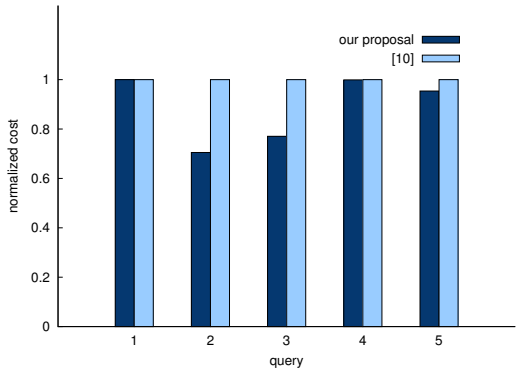
Fig. 15 Normalized cost for evaluating different queries under different authorization profiles

plaintext some of the attributes that were only accessible in encrypted form in $\mathbf{UA_{mix}P_{enc}}$ scenario.

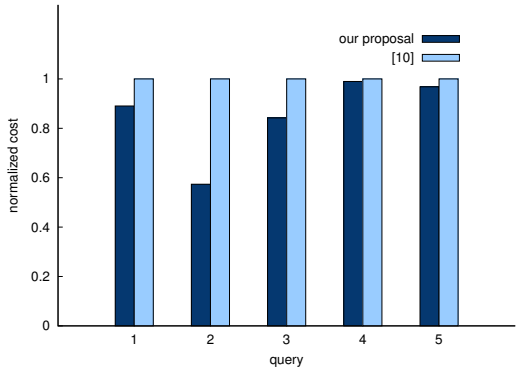
These scenarios enable increasing involvement of authorities and computational providers in performing computation, hence leveraging authorizations supported by our model for delegating computational intensive portions of the query and enjoy cost-saving opportunities. The baseline for comparison is given by considering the case, which we denote as **U**, in which the query is executed completely by the user itself.

Cost configurations and economic benefits. We set the cost values input to the experiments considering, as it is to be expected in the scenarios that motivate this research, a relatively high cost for the direct involvement of the user and of data authorities. In particular, also based on considerations from our use-case and on the listings of the most common cloud providers on the market (e.g., Amazon AWS, Google Cloud Platform), we assumed the cpu usage and data transfer costs of the user from 2 to 10 times that of data authorities, and from 20 to 100 times that of cloud providers. We then performed different experiments for different cost combinations, having confirmation from each of them of cost-saving in the adoption of our approach. Figures 15(a-b) illustrate the results in the two configurations at the extreme of our considered cost ranges where cost-saving is lowest (i.e., 2-20 scenario) and highest (i.e., 10-100 scenario). Given the heterogeneity of the different queries and their cost, we report the cost in a normalized form considering, for each query, a unitary cost for **U** (reported by the continuous horizontal line in the figure). As expected, compared with the base scenario (**U**) where only the user can perform computations, the involvement of data authorities and providers enables significant savings. Indeed, our approach permits to partially delegate operations running on encrypted data to cloud providers with economically convenient price lists, even if they are not trusted to access plaintext data. As it is to be expected, the more permissive the authorizations, the larger the potential savings, which reach already considerable levels when providers are allowed to access data only in encrypted form. As a matter of fact, as visible from the figure, for the 2-20 configuration cost reduction ranges and from 69% (q_1 in $\mathbf{UA_{mix}P_{enc}}$) to 94% (q_4 in $\mathbf{UA_{mix}P_{mix}}$). For the 10-100 configuration, cost reduction ranges from 92% (q_1 in $\mathbf{UA_{mix}P_{enc}}$) to 98% (q_4 in $\mathbf{UA_{mix}P_{mix}}$). The higher cost reductions in the 10-100 configuration, with respect to the 2-20 configuration, are clearly due to the impact of the difference in cost savings (higher in the former and lower in the latter), which has its effect throughout query execution. The reason for the different cost reductions can be interpreted observing that in q_4 and q_5 (which show a higher cost reduction) the udf calls are closer to the leaves (and therefore operate on more data), while in q_2 and q_3 udf calls are closer to the root. Also, q_1, which sees a lower (with respect to the other queries) cost reduction for the 2-20 configuration, involves a lower computational effort (q_1 does not call any udf).

We close this section with a note on the improvement brought by the constraint-modeling formulation presented with respect to the modeling in [10], where operation assignment did not take into account the cost of encryption and decryption operations that the assignment would have entailed. Considering such cost in computing the assignment allows us to rule out a solution if the cost needed for encryption/decryption would eventually make it more expensive than alternatives. Figure 16 compares the (normalized) costs of the extended query plans generated by our approach for the considered five queries and the ones computed according to the heuristics in [10]. The figure has been obtained considering the $\mathbf{UA_{mix}P_{mix}}$ authorization scenario, since it is the most general authorization scenario. As expected, in the tested configurations, our assignment presents higher economic benefits compared



(a) 2-20 configuration



(b) 10-100 configuration

Fig. 16 Normalized cost for evaluating different queries with our solution and the solution in [10]

with the assignment computed by the heuristics in [10] (up to 29% for the 2-20 configuration and up to 42% for the 10-100 configuration). In both configurations, the higher cost reductions observable in q_2 and q_3 are motivated by the fact that these queries see the involvement of (and return) more data – compared to the other queries – which need to be encrypted and subsequently decrypted for enabling the involvement of the providers chosen by the heuristics in [10]. The presented constraint-modeling formulation is able to consider these costs, hence producing a different assignment that, while seeing the involvement of providers that are not the most economic for computation and transfer costs, require less encryption and decryption, ensuring a reduction in the cost for the overall execution.

9 Related work

The problem of managing queries in distributed scenarios has been extensively studied, but traditional solutions (e.g., [19,21]) as well as modern approaches that consider big data analytics (e.g., [2,4,25]) do not take into consideration access restrictions. In the relational database context, access restrictions can be supported by views (e.g., [9,17,26]), access patterns (e.g., [3,6]), or data masking (e.g., [20]). Such proposals however do not consider encryption.

Work closest to ours has addressed the problem of protecting data confidentiality in distributed computations (e.g., [11,22,27,32]). In [32] the authors present an approach to collaboratively execute queries on data subject to access restrictions, considering different join evaluation strategies. In [27] the authors propose an operator placement approach aimed at satisfying privacy constraints, while maximizing performance in query evaluation. The proposed solution relies on programming language techniques for regulating and controlling information flows. In [11] the authors provide a solution for restricting access and sharing of distributed data, which supports the explicit consideration of join paths in the authorizations. The proposal in [22] aims at protecting computations in hybrid clouds, preventing flows of sensitive information to the public cloud. These works confirm the relevance of the problem, but focus on different aspects. In particular, the approach in [32] considers only data explicitly exchanged among providers and do not take into consideration implicit information disclosure. While providing a more expressive authorization model, the approach in [11] requires collaborative specification of authorizations. None of the proposals considers the possibility of protecting data with encryption. Our proposal takes then a novel approach supporting different visibility levels over data and flexibly injecting encryption on-the-fly to protect data and enable the controlled involvement of cloud providers in query computation. In [15] the authors address a complementary problem allowing users to specify confidentiality requirements in query evaluation to protect the objective of their queries to some providers. The idea of specifying different visibility levels over data has been first proposed in [10]. The approach in [14] integrates this authorization model in a distributed query optimizer. In this paper, we considerably extend the prior work in [10] by enriching it with the support for additional operators (i.e., rename and set operators), providing formal proofs of theorems, and introducing a novel approach for identifying a minimum cost assignment, also taking into consideration the cost of encryption and decryption operations.

Other related work has investigated leveraging Trusted Execution Environments (e.g., Intel SGX) for storing or processing sensitive data, in an otherwise non fully trusted scenario or host [24,28,31]. These works are complementary to ours and the consideration of trusted execution environments for delegating part of

the computation in our model can represent an interesting direction of investigation.

Several works (e.g., [1,18,23,29]) have investigated the use and support of encryption for the protection of data in storage or query execution. Other approaches (e.g., [5,7]) proposed solutions for using secure multiparty computation in query evaluation, to keep both the input operands and the result secret to the party in charge of query evaluation. Specific works (e.g., [12]) have designed techniques to verify the integrity of query results computed by potentially untrusted providers. All these solutions are complementary to our proposal.

10 Conclusions

We leverage the availability of emerging solutions supporting computation over encrypted data to provide a novel flexible approach enabling controlled query execution in the cloud. Our approach allows independent data authorities to make their data available for access and collaborative query execution, and enables users to execute queries over such data with selective and controlled involvement of external cloud providers. A main advantage of our approach is the flexibility in the assignment of query operations to providers as most economically convenient, with on-the-fly insertion of encryption and decryption to adjust visibility of data as dictated by the authorizations. The experimental evaluation confirms the benefits provided by our proposed authorization model. Our work leaves room for extensions, among which we mention two in particular. The first extensions can be the inclusion in the authorization specification of encryption schemes. In other words, authorizations could support different encryption 'levels', for instance limiting encrypted visibility only to certain 'types' of encryption. With respect to its impact on the model, this would imply the non validity of encrypted visibility (i.e., ineffectiveness of authorizations on an encrypted attribute) if the encryption required for the execution of the operation is not allowed by the authorization. A second interesting direction of investigation is the consideration of Trusted Execution Environments. Intuitively, TEEs on board of computational providers can be captured assuming the presence of an additional provider, modeling the trusted execution environment, characterized by its own set of authorizations and price list. However, the authorizations of the TEE and the computational provider hosting it cannot be completely independent since communication to the TEE passes through the hosting environment. Basically, the authorizations of a TEE should be at least as permissive as the ones of its hosting environment, and as a matter of fact more permissive (e.g., a TEE can be authorized a plaintext access to attributes that its hosting environment can view only in encrypted form). However, since communication to the TEE passes through the hosting environment, attributes going to the TEEs need to be accessible (even if in more restrictive form) to the hosting environment. With respect to operation assignments, computational cost to be considered is the one of the TEE and the communication cost is the one of its hosting environment.

References

1. Agrawal, R., Asonov, D., Kantarcioglu, M., Li, Y.: Sovereign joins. In: Proc. of ICDE. (2006)
2. Alkowaileet, W., Alsubaiee, S., Carey, M., Li, C., Ramampiaro, H., Sinthong, P., Wang, X.: End-to-end machine learning with Apache AsterixDB. In: Proc. of DEEM. (2018)
3. Amarilli, A., Benedikt, M.: When can we answer queries using result-bounded data interfaces? In: Proc. of PODS. (2018)
4. Armbrust, M., Xin, R.S., Lian, C., Huai, Y., Liu, D., Bradley, J.K., Meng, X., Kaftan, T., Franklin, M.J., Ghodsi, A., Zaharia, M.: Spark SQL: Relational data processing in Spark. In: Proc. of SIGMOD. (2015)
5. Bater, J., Elliott, G., Eggen, C., Goel, S., Kho, A., Duggan, J.: SMCQL: Secure query processing for private data networks. PVLDB **10**(6), 673–684 (2017)
6. Benedikt, M., Leblay, J., Tsamoura, E.: Querying with access patterns and integrity constraints. PVLDB **8**(6), 690–701 (2015)
7. Chow, S.S., Lee, J.H., Subramanian, L.: Two-party computation model for privacy-preserving queries over distributed databases. In: Proc. of NDSS. (2009)
8. Cohen, J., Dolan, B., Dunlap, M., Hellerstein, J.M., Welton, C.: Mad skills: New analysis practices for big data. PVLDB **2**(2), 1481–1492 (2009)
9. De Capitani di Vimercati, S., Foresti, S., Jajodia, S., Livraga, G., Paraboschi, S., Samarati, P.: Fragmentation in presence of data dependencies. IEEE TDSC **11**(6), 510–523 (2014)
10. De Capitani di Vimercati, S., Foresti, S., Jajodia, S., Livraga, G., Paraboschi, S., Samarati, P.: An authorization model for multi-provider queries. PVLDB **11**(3), 256–268 (2017)
11. De Capitani di Vimercati, S., Foresti, S., Jajodia, S., Paraboschi, S., Samarati, P.: Authorization enforcement in distributed query evaluation. JCS **19**(4), 751–794 (2011)
12. De Capitani di Vimercati, S., Foresti, S., Jajodia, S., Paraboschi, S., Samarati, P.: Efficient integrity checks for join queries in the cloud. JCS **24**(3), 347–378 (2016)
13. De Capitani di Vimercati, S., Foresti, S., Livraga, G., Samarati, P.: Practical techniques building on encryption for protecting and managing data in the cloud. In: P. Ryan, D. Naccache, J.J. Quisquater (eds.) Festschrift for David Kahn, pp. 205–239. Springer (2016)
14. Dimitrova, E., Chrysanthis, P., Lee, A.: Authorization-aware optimization for multi-provider queries. In: Proc. of SAC. (2019)
15. Farnan, N., Lee, A., Chrysanthis, P., Yu, T.: PAQO: Preference-aware query optimization for decentralized database systems. In: Proc. of ICDE. (2014)

16. Grofig, P., Haerterich, M., Hang, I., Kerschbaum, F., Kohler, M., Schaad, A., Schroepfe, A., Tighzert, W.: Experiences and observations on the industrial implementation of a system to search over outsourced encrypted data. In: Proc. of Sicherheit. (2014)
17. Guarnieri, M., Basin, D.: Optimal security-aware query processing. PVLDB **7**(12), 1307–1318 (2014)
18. Hacigümüs, H., Iyer, B., Mehrotra, S., Li, C.: Executing SQL over encrypted data in the database-service-provider model. In: Proc. of SIGMOD. (2002)
19. Kossmann, D.: The state of the art in distributed query processing. ACM CSUR **32**(4), 422–469 (2000)
20. Kwakye, M.M., Barker, K.: Privacy-preservation in the integration and querying of multidimensional data models. In: Proc. of PST. (2016)
21. Levy, A.Y., Srivastava, D., Kirk, T.: Data model and query evaluation in global information systems. JIIS **5**(2), 121–143 (1995)
22. Oktay, K.Y., Kantarcioglu, M., Mehrotra, S.: Secure and efficient query processing over hybrid clouds. In: Proc. of ICDE. (2017)
23. Popa, R., Redfield, C., Zeldovich, N., Balakrishnan, H.: CryptDB: Protecting confidentiality with encrypted query processing. In: Proc. of SOSP. (2011)
24. Priebe, C., Vaswani, K., Costa, M.: EnclaveDB: A secure database using SGX. In: Proc. of SP. (2018)
25. Rheinländer, A., Leser, U., Graefe, G.: Optimization of complex dataflows with user-defined functions. ACM CSUR **50**(3), 38:1–38:39 (2017)
26. Rizvi, S., Mendelzon, A., Sudarshan, S., Roy, P.: Extending query rewriting techniques for fine-grained access control. In: Proc. of SIGMOD. (2004)
27. Salvaneschi, G., Köhler, M., Sokolowski, D., Haller, P., Erdweg, S., Mezini, M.: Language-integrated privacy-aware distributed queries. Proc. ACM Program. Lang. **3**(OOPSLA) (2019)
28. Thoma, C., Lee, A., Labrinidis, A.: Behind enemy lines: Exploring trusted data stream processing on untrusted systems. In: Proc. of CODASPY. (2019)
29. Tu, S., Kaashoek, M., Madden, S., Zeldovich, N.: Processing analytical queries over encrypted data. PVLDB **6**(5), 289–300 (2013)
30. Vaidya, J.: Privacy in the context of digital government. In: Proc. of DG.O. (2012)
31. Vinayagamurthy, D., Gribov, A., Gorbunov, S.: StealthDB: a scalable encrypted database with full SQL query support. PoPETS **2019**(3), 370–388 (2019)
32. Zeng, Q., Zhao, M., Liu, P., Yadav, P., Calo, S., Lobo, J.: Enforcement of autonomous authorizations in collaborative distributed query evaluation. IEEE TKDE **27**(4), 979–992 (2015)

A Proofs of theorems

Theorem 1 *Let* `T(N)` *be a query tree plan.* $\forall n_x, n_y \in$ `N` *with profile* $[R_x^{vp}, R_x^{ve}, R_x^{ip}, R_x^{ie}, R_x^{\simeq}]$ *and* $[R_y^{vp}, R_y^{ve}, R_y^{ip}, R_y^{ie}, R_y^{\simeq}]$*, respectively, s.t.* n_y *is a descendant of* n_x*:*

i) $(R_y^{vp} \cup R_y^{ve} \cup R_y^{ip} \cup R_y^{ie} \cup \{A | A \in R_y^{\simeq}\}) \subseteq (R_x^{vp} \cup R_x^{ve} \cup R_x^{ip} \cup R_x^{ie} \cup \{A | A \in R_x^{\simeq}\})$

ii) $\forall A \in R_y^{\simeq}, \exists A' \in R_x^{\simeq}, A \subseteq A'$.

Proof We separately prove the two conditions of the theorem.

i) Let us first analyze the case in which n_x is the direct ancestor of n_y. Assume, by contradiction, that $\exists a \in \{R_y^{vp} \cup R_y^{ve} \cup R_y^{ip} \cup R_y^{ie} \cup \{A | A \in R_y^{\simeq}\}\}$ s.t. $a \notin \{R_x^{vp} \cup R_x^{ve} \cup R_x^{ip} \cup R_x^{ie} \cup \{A | A \in R_x^{\simeq}\}\}$. This would imply that attribute a is removed from the profile of R_x by the execution of the operation represented by n_x. According to the operations in Figures 2 and 3, projection, group-by, udf, and rename operations remove attributes from relation profiles (and, more precisely, from the visible components of profiles). However, the attributes removed from the visible components by rename operation are inserted into the renamed attributes component and, from there, into the components of the relation profile where the new attribute name appears. The attributes removed from the visible components by projection, group-by, and udf operations already belong to $R_x^{ip} \cup R_x^{ie} \cup \{A | A \in R_x^{\simeq}\}$. In fact, since projections have been pushed down in `T(N)`, the first projection removes all attributes that are neither involved in operations in the query plan, nor returned in the query result. Therefore, for each relation, only the attributes explicitly appearing in the clauses of the query survive in the profile of the relation corresponding to the projection pushed down at each relation. The attributes removed can only be the attributes on which operations have already been evaluated, since otherwise the query could not be evaluated correctly. The operations in which an attribute a, removed by the projection, the group-by, or the udf at n_x, have possibly been involved (as illustrated in Figure 2) are: selection (a would be in R_x^{ip}, R_x^{ie}, or $R_x^{\simeq}$); join (a would be in $R_x^{\simeq}$); group-by (a would be in R_x^{ip} or R_x^{ie}); a set operator (a would be in $R_x^{\simeq}$); and udf (a would be in $R_x^{\simeq}$). Note that the cartesian product does not specifically operate on any attribute. Also, attributes involved in aggregations will be subject to operations or will belong to the query result. Encryption/decryption operations are instead functional to query evaluation. Hence, no attribute is removed from the profile of R_x, contradicting our hypothesis.
Since $\forall n_x, n_y$ s.t. n_y is a direct descendant of n_x, $(R_y^{vp} \cup R_y^{ve} \cup R_y^{ip} \cup R_y^{ie} \cup \{A | A \in R_y^{\simeq}\}) \subseteq (R_x^{vp} \cup R_x^{ve} \cup R_x^{ip} \cup R_x^{ie} \cup \{A | A \in R_x^{\simeq}\})$, by the transitivity of operator $\subseteq$ the first condition of the theorem holds.

ii) Let us first analyze the case in which n_x is the direct ancestor of n_y and assume, by contradiction, that $\exists A \in R_y^{\simeq}$ s.t. $\nexists A' \in R_x^{\simeq}, A \subseteq A'$. The sets of attributes included in $R_y^{\simeq}$ are impacted only when the operation in n_x is one of the operations described in the following.

ii.1) n_x is a cartesian product. The cartesian product combines $R_y^{\simeq}$ with $R_z^{\simeq}$, with R_z the other operator of n_x (i.e., $R_x^{\simeq} = R_y^{\simeq} \cup R_z^{\simeq}$). Then, if $A \in R_y^{\simeq}$ and $\exists A_i \in R_z^{\simeq}$ s.t. $A \cap A_i \neq \emptyset$, then $A' = A \cup A_i$ is inserted into $R_x^{\simeq}$ in place of A. Otherwise, A belongs to the $R_x^{\simeq}$. This contradicts our hypothesis.

ii.2) n_x is a selection or join with condition a_i *op* a_j. The selection/join operations cause $R_x^{\simeq} = R_y^{\simeq} \cup R_z^{\simeq} \cup \{a_i, a_j\}$, which inserts equivalence $\{a_i, a_j\}$ in the result of $R_y^{\simeq} \cup R_z^{\simeq}$. Then, it merges the set $A_i \in (R_y^{\simeq} \cup R_z^{\simeq})$ s.t. $a_i \in A_i$ with the set $A_j \in (R_y^{\simeq} \cup R_z^{\simeq})$ s.t. $a_j \in A_j$, producing a new set $A_{ij} = A_i \cup A_j$, if such sets exist; it inserts a_j into A_i if A_j does not exist (and viceversa), producing a new set $A_{ij} = A_i \cup \{a_j\}$ (or $A_{ij} = A_j \cup \{a_i\}$) in place of A_i or A_j, respectively. It creates set $A_{ij} = \{a_i, a_j\}$ if neither A_i nor A_j exist. The set $R_x^{\simeq}$ is then obtained as $R_x^{\simeq} = R_y^{\simeq} \cup R_z^{\simeq} \setminus \{A_i, A_j\} \cup \{A_{ij}\}$. Therefore, if $a_i \notin A$ and $a_j \notin A$ (remember that $A \in R_y^{\simeq}$), then $A \in R_x^{\simeq}$. Otherwise, $A_{ij} \in R_x^{\simeq}$ and $A \subset A_{ij}$. This contradicts our hypothesis.

ii.3) n_x is a set operator. Any set operator causes $R_x^{\simeq} = R_y^{\simeq} \cup R_z^{\simeq} \cup \{a_{yi}, a_{zi}\}$, which inserts equivalence $\{a_{yi}, a_{zi}\}$, for $i = 1, \ldots, |R_y^{vp} \cup R_y^{ve}|$, in the result of $R_y^{\simeq} \cup R_z^{\simeq}$. The insertion of each pair $\{a_{yi}, a_{zi}\}$ into the result of $R_y^{\simeq} \cup R_z^{\simeq}$ operates as illustrated above for the selection/join operation. Hence, if

$a_{yi} \notin A$ and $a_{zi} \notin A$ (remember that $A \in R_y^{\simeq}$), then $A \in R_x^{\simeq}$, else $A_{yizi} \in R_x^{\simeq}$ and $A \subset A_{yizi}$. This contradicts our hypothesis.

ii.4) n_x is a udf operating over a set A_x of attributes. The udf operation causes $R_x^{\simeq} = R_y^{\simeq} \cup A_x$, which inserts equivalence A_x into $R_y^{\simeq}$. Then, it merges the set $A_i \in R_y^{\simeq}$ s.t. $A_i \cap A_x \neq \emptyset$ with the set A_x, producing a new set $A_{ix} = A_i \cup A_x$, if such set exists, and inserts A_{ix} into $R_y^{\simeq}$ in place of A_i. It creates set A_x otherwise. Therefore, if $A_x \cap A = \emptyset$, then $A \in R_x^{\simeq}$. Otherwise, $A_{ix} \in R_x^{\simeq}$ and $A \subset A_{ix}$. This contradicts our hypothesis.

Renaming does not have impact on the second condition of the theorem, since renamed attributes are substituted by the corresponding original attribute names when the profile is closed (Definition 3). Since $\forall n_x, n_y$ s.t. n_y is a direct descendant of n_x, $\forall A \in R_y^{\simeq}, \exists A' \in R_x^{\simeq}$ s.t. $A \subseteq A'$, for the transitivity of operator $\subseteq$, the second condition of the theorem holds. □

Theorem 2 *Let* `T(N)` *be a query tree plan,* $n \in$ `N` *be a non-leaf node* $n_l, n_r \in$ `N` *be its non-leaf children, if any.* $\hat{R}_l^{vp} \cup \hat{R}_r^{vp} \subseteq \hat{R}^{ip} \Longrightarrow \Lambda(n_x) \subseteq \Lambda(n), \forall n_x$ *ancestor of* n.

Proof Let us first analyze the case in which n_x is the direct ancestor of n in `T(N)` and assume, by contradiction, that $\exists S \in \Lambda(n_x)$ s.t. $S \notin \Lambda(n)$. By Definition 8, this implies that S is authorized for relation R_x produced by n_x over operands $\hat{R}$ and possibly $\hat{R}_w$, with n_w the other direct descendant of n_x if n_x represents a binary operation, and S is authorized for $\hat{R}$ and $\hat{R}_w$ (if it is the case). At the same time, S is not authorized for R, $\hat{R}_l$, and/or $\hat{R}_r$. By Theorem 1, all attributes in the profile of a node also belong to the profiles of its ancestors. Then, S could be authorized for R_x and not for R only if there exists an attribute $a \in \mathcal{E}_S$ s.t. a appears plaintext (visible and/or implicit) in the profiles of $\hat{R}_l$, $\hat{R}_r$, or R and is included encrypted in the profiles of $\hat{R}$, $\hat{R}_w$, and R_x. Let us separately analyze the cases in which a is visible plaintext and implicit plaintext. If a appears implicit plaintext in the profile of R or of an operand of n (meaning in $\hat{R}_l$ or $\hat{R}_r$), since no operation removes attributes from an implicit component of a profile (see Figure 2), then a will also be included in the implicit plaintext component of the profiles of all ancestors of n, including n_x. Therefore, $S \notin \Lambda(n_x)$, contradicting our hypothesis. Let us now analyze the case in which a is visible plaintext in the profile of $\hat{R}_l$, $\hat{R}_r$, or R. In all these cases, by Definition 7, a is needed plaintext for the execution of the operation in n (as otherwise it would be encrypted in $\hat{R}_l$, $\hat{R}_r$, and then also in the profile of the relation resulting from n). However, by hypothesis $\hat{R}_l^{vp} \cup \hat{R}_r^{vp} \subseteq \hat{R}^{ip}$. Then, a would be included in the implicit plaintext components of the ancestors of n, thus making $S \notin \Lambda(n_x)$, contradicting our hypothesis. Since $\forall n, n_x$ s.t. n_x is the direct ancestor of n, $\hat{R}_l^{vp} \cup \hat{R}_r^{vp} \subseteq \hat{R}^{ip} \Longrightarrow \Lambda(n_x) \subseteq \Lambda(n)$, for the transitivity of operator $\subseteq$, the theorem holds. □

Theorem 3 *Let* `T(N)` *be a query plan, and* Λ *be a candidate assignment function for it:*

i) $\forall$ `T'` $\in \mathcal{T}$, λ, *and* $n \in$ `N`, *if* `T'` *is an extended query plan for* `T` *and* λ *is an authorized assignment for* `T'`, *then* $\lambda(n) \in \Lambda(n)$.

ii) $\forall \lambda$, *if* $\forall n \in$ `N`, $\lambda(n) \in \Lambda(n)$, *then there exists an extended query plan* `T'` *for* `T` *such that* λ *is an authorized assignment for* `T'`.

Proof To clearly distinguish between nodes of the original tree `T(N)` and the same nodes in the extended tree `T'(N)`, we will denote with n' the counterpart in `T'(N)` of node n in `T(N)`. We now separately prove the two conditions of the theorem.

i) Suppose, by contradiction, that $\exists S = \lambda(n')$ s.t. $S \notin \Lambda(n)$, meaning that S is authorized for n', n'_l, and n'_r and not for n, $\hat{R}_l$, and $\hat{R}_r$. This can occur in two scenarios.

i.1) $\exists a$ in the profiles of n, $\hat{R}_l$, and/or $\hat{R}_r$ s.t. a does not belong to the profiles of n', n'_l, and n'_r, and $a \notin \mathcal{P}_S \cup \mathcal{E}_S$.
Theorem 1 states that all attributes in the profile of a relation belong to the profile of its ancestor. Therefore, if a does not belong to the profile of n (n', resp.), then a belongs to the profiles of neither n_l nor n_r (n'_l nor n'_r, resp.). On the other hand, if a belongs to the profile of n (n', resp.), then a certainly belongs to the profiles of either n_l or n_r (n'_l or n'_r, resp.). Therefore, we can focus on the profiles of n and n'. The profile of n is computed assuming operands $\hat{R}_l$ and $\hat{R}_r$. According to Definition 7, the computation of minimum required views does not change which attributes are included in the profile of a node. This implies that the attributes in the profile of n be the same of n', contradicting our hypothesis.

i.2) $\exists a$ appearing plaintext in the profiles of n, $\hat{R}_l$, and/or $\hat{R}_r$ s.t. a is encrypted in the profiles of n', n'_l, and n'_r, and $a \in \mathcal{E}_S$.
Let us first analyze the case in which a is visible plaintext in the profile of n, $\hat{R}_l$, and/or $\hat{R}_r$. In all these cases, by Definition 7, a is needed plaintext for the execution of the operation in n but then it should also be represented in the clear also in n', n'_l, and n'_r to ensure computability of the operation, thus contradicting our hypothesis.
Let us now analyze the case in which a is implicit plaintext in the profile of n, $\hat{R}_l$, or $\hat{R}_r$. This can occur only if an operation over a has been executed by (at least) one descendant n_d of n and left a trace in the implicit component. Since n_d, being in `T(N)`, operates on the minimum required view(s) of its descendant(s), it left a trace in the implicit plaintext component of the profile of n_d only if the operation required to operate on the plaintext representation of a. However, the same operation is to be evaluated also by n'_d in `T'(N)`, and therefore a appears in the implicit plaintext component of the profiles of n', n'_l, and/or n'_r, thus contradicting our hypothesis.

ii) Suppose, by contradiction, that $\forall n, S = \lambda(n) \in \Lambda(n)$ and that $\nexists$ `T'(N)` s.t. `T'(N)` is an extended plan for `T(N)` for which λ is an authorized assignment. This can occur in two scenarios.

ii.1) $\exists a$ in the profiles of n', n'_l, and/or n'_r s.t. a does not belong to the profiles of n, $\hat{R}_l$, $\hat{R}_r$, and $a \notin \mathcal{P}_S \cup \mathcal{E}_S$.
As previously shown, the sets of attributes in the profile of a node in `T(N)` and of its counterpart in `T'(N)` include the same set of attributes.

ii.2) $\exists a$ plaintext in the profiles of n', n'_l, and/or n'_r s.t. a is encrypted in the profiles of n, $\hat{R}_l$, and $\hat{R}_r$, and $a \in \mathcal{E}_S$.
Let us first analyze the case in which a is visible plaintext in the profiles of n', n'_l, and/or n'_r. Since a appears in encrypted form in the profiles of the original query plan, plaintext visibility over a is not required to execute the operation in n'. Then, `T'(N)` can be extended encrypting a before n'.
Let us now analyze the case in which a is implicit plaintext in the profiles of n', n'_l, and/or n'_r. In this case, an operation inserting a into the implicit component of a profile has been carried out over the plaintext representation of a in (at least) one descendant n'_d of node n' in `T'(N)`. However, since a belongs to the implicit plaintext component of the profiles of neither $\hat{R}_l$, nor $\hat{R}_r$, this operation can also be evaluated over the encrypted representation of a. Hence, `T'(N)` can be extended with an encryption operation over a preceding n'_d. This includes a in the implicit encrypted component in the profile of n'_d and of its ancestors, rather than their implicit plaintext component. Indeed, no operation moves attributes out from implicit components (see Figures 2 and 3). □