101017171 — MARSAL — H2020-ICT-2020-2

Deliverable D5.2

Initial report on data privacy protocols in multi-tenant infrastructures

Document Summary Information

Grant Agreement No	101017171	**Acronym**	MARSAL
Full Title	Machine Learning-based, Networking and Computing Infrastructure Resource Management of 5G and Beyond Intelligent Networks		
Start Date	01/01/2021	**Duration**	36 months
Project URL	`https://www.marsalproject.eu/`		
Deliverable	D5.2 - Initial report on data privacy protocols in multi-tenant infrastructures		
Work Package	WP5		
Contractual due date	M18	**Actual submission date**	2022.06.30
Nature	Report	**Dissemination Level**	Public
Lead Beneficiary	UNIMI		
Main editors	Sabrina De Capitani di Vimercati (UNIMI), Pierangela Samarati (UNIMI)		
List of contributors	Sabrina De Capitani di Vimercati (UNIMI), Sara Foresti (UNIMI), Giovanni Livraga (UNIMI), Vincenzo Piuri (UNIMI), Pierangela Samarati (UNIMI)		

Revision history

Version	Issue Date	Changes	Contributor(s)
0.1	2022.05.22	First draft	Sabrina De Capitani di Vimercati (UNIMI), Sara Foresti (UNIMI), Giovanni Livraga (UNIMI), Vincenzo Piuri (UNIMI), Pierangela Samarati (UNIMI)
0.2	2022.06.09	Second draft	Sabrina De Capitani di Vimercati (UNIMI), Sara Foresti (UNIMI), Giovanni Livraga (UNIMI), Vincenzo Piuri (UNIMI), Pierangela Samarati (UNIMI)
0.3	2022.06.19	Third draft	Sabrina De Capitani di Vimercati (UNIMI), Sara Foresti (UNIMI), Giovanni Livraga (UNIMI), Vincenzo Piuri (UNIMI), Pierangela Samarati (UNIMI)
1.0	2022.06.30	Final version	Sabrina De Capitani di Vimercati (UNIMI), Sara Foresti (UNIMI), Giovanni Livraga (UNIMI), Vincenzo Piuri (UNIMI), Pierangela Samarati (UNIMI)

Disclaimer

Copyright message

Table of Contents

List of Acronyms and Abbreviations

ACRONYM	DESCRIPTION
A	APid (identifier of an access point in relation AP)
AP	Access Point
C	Cid (identifier of relation CDR)
CDR	Call Detail Record
DC	Data Center
DCS	Distributed Cloud Storage
ENC_DEC	Encryption/Decryption cost
H	HashId (hash of the identifier of a user device in relation CDR)
I	IdAP (identifier of an access point in relation CDR)
N	NumAntennas (number of antennas of an access point in relation AP)
OP_EXEC	Operation Execution cost
P	PLatitude (latitude of an access point in relation AP)
M	MLongitude (longitude of an access point in relation AP)
T	Time (time of a device action in relation CDR)
TRANSF	Data Transfer cost
UDF	User Defined Function

List of Figures

List of Tables

Executive Summary

5G and beyond networks have been bringing tremendous changing to our society, offering new and enhanced applications and services that have been profoundly impacting how we live, work, and engage with the environment surrounding us. The huge opportunities and benefits introduced by the advancements of the network infrastructure and its enabled hyper-connected society, can be hampered by security and privacy concerns. Indeed, the (actual or perceived) loss of control over data and applications in such complex interconnected scenarios can have a strong detrimental impact on the full realization of the potential in front of us.

MARSAL aims at addressing the security and privacy concerns in novel scenarios enabled by 5G and beyond networks, and providing effective solutions for them, through a dedicated Work Package, WP5 on "Security in Multi-Tenant Infrastructures". The investigation in WP5 is organized in different tasks, aimed respectively at the development of: a decentralized framework for confidentiality and direct contracts in B5G networks (T5.1), policy-driven data integrity and privacy protocols in multi-tenant infrastructures (T5.2), a ML-assisted, hardware accelerated slice security mechanisms (T5.3), and the implementation, integration and testing of MARSAL's security mechanisms for multi-tenant infrastructures (T5.4).

Within WP5, T5.2 (policy-driven data integrity and privacy protocols in multi-tenant infrastructures) is concerned with the protection of data in multi-tenant infrastructures and in the development of a policy-driven solution for data security and privacy and accompanying techniques for its enforcement. Data, and therefore their protection, play in fact a major role in novel scenarios enabled by advanced infrastructures, they are at the very heart of their own working and at the basis of all advanced applications, with data generated by distributed devices and sensors, and gathered, processed and shared at a face pace and with ever growing computing and analytics capacity. The goal of the task is then to enable the control on data as they are distributed and processed in the infrastructure, to ensure their integrity and confidentiality, with a policy-driven approach enabling the specification of protection needs and calling upon underlying techniques for their enforcement.

This deliverable reports on the investigation carried out and the findings in the first period of the project within T5.2. In particular, it summarizes the investigation on the security challenges and problems and presents the preliminary design for a policy-driven approach enabling controlled secure collaboration among different nodes in the infrastructure for performing distributed collaborative computations involving data under control of different parties and tenants. Chapter 1 then summarizes the work done in the first period of the project, and discusses the state-of-the-art related to the distributed processing on which the deliverable is focused, highlighing the main innovation introduced by MARSAL. Chapter 2 presents a solution for supporting different infrastructure owners in the definition of the access restrictions that must be considered when accessing and processing their data. The Chapter also illustrates how such restrictions can be enforced when executing a computation. Chapter 3 describes a solution for assigning the operations of a computation to subjects in such a way that a parameter of interest, in our case the cost of the overall computation, is minimized. The assignment problem is modeled as a binary programming problem that can be then solved with off-the-shelf solvers. Chapter 4 concludes the deliverable.

1. Introduction

5G and beyond networks are a key enabler of today's digital society, supporting new and better applications in a variety of sectors thank you to the availability of a powerful hyperconnected infrastructure offering unprecedented network capacity and speed. Distributed sensors, mobile and pervasive devices, cloud/edge/fog computational and storage nodes, can be involved in providing advanced services and applications. At the center of such novel applications are data, gathered, generated, shared, processed and communicated among the different components of the infrastructure at an incredible pace. Such data can be private, sensitive, or company-confidential. At the same time, different components of the infrastructure might be under different administrative domains and subject to different trust assumptions.

The full realization of the power brought by the hyperconnected infrastructure can happen only with the availability of solutions to ensure proper protection of the data across their whole life cycle in the infrastructure. Task T5.2 (started at M9) aims at addressing such data protection issues and at designing novel security services to enforce effective data protection in multi-tenants environments.

In this chapter, we summarize the goal of the task and the investigation carried out in the first period of the project.

1.1 Mapping MARSAL outputs

We have investigated different challenges, approaches, and open issues related to ensuring proper security and privacy in multi-tenants infrastructures and their enabled novel scenarios such as smart cities that are characterized by massive data growth and processing requirements [18]. For instance, intelligent transportation can leverage mobility data coming from connected vehicles, images coming from surveillance cameras, information on accidents and plans for road works to support commuters in finding the best way to move from home to work and vice-versa, with clear environmental benefits in terms of reduced traffic congestion and emissions. The storage and processing requirements can be carried out through the use of a variety of networking resources as well as edge/cloud computing technologies and multi-tenant infrastructures that are under the control of different administrative domains (providers). Challenges here relate to the need of guaranteeing the collaboration among such different providers, without assuming trust among them, for supporting data sharing and processing in a way that no improper access or flow is enacted. Data sanitization and obfuscation can also be provided to ensure private or sensitive information gathered from devices is protected. In this context, we have worked on the identification and specification of protection requirements, aiming at understanding the different aspects of protection that may need to be taken into consideration when involving and relying on external (not fully trusted) parties in the storage or processing of data [19].

We have also investigated how the availability of high performance networks can help in the development of efficient data anonymization solutions for massive amount of data, through the distribution of data anonymization tasks to different nodes in the network. A challenge here is how to perform data fragmentation and distribution among different nodes in the network, each then operating with partial knowledge of the data collection, while ensuring goodness (in terms of utility) of the overall anonymization result [9, 11]. Related to this, there is also the problem of enabling fine-grained access and retrieval of data that are stored in encrypted form at the different nodes of the infrastructure. We have then proposed a multi-dimensional index (i.e., an index on multiple attributes) that is robust against inference exposure and, at the same time, performs well for data retrieval [10].

GA component title	GA component outline	Respective document section(s)	Justification
Task T5.2	Task T5.2 includes the definition of a policy-based approach offering different degree of visibility over data as well as sharing and processing restrictions in computations involving multiple datasets.	Chapter 2 and Chapter 3	These Chapters describe a policy-based approach supporting the controlled execution of computations over multiple datasets. The proposed approach regulates the (cross-domain) flow of data, depending on the parties involved in the computation.
Deliverable			
Deliverable D5.2 on "Initial report on data privacy protocols in multi-tenant infrastructures" describes the first results obtained in the context of task T5.2 that focuses on data integrity and privacy solutions.			

Table 1.1: MARSAL outputs

The main core of our investigation has addressed the development of a policy-driven solution for protecting data in multi-tenants infrastructure [14] (see Table 1.1). The goal is to enable the controlled involvement of different parties in the network infrastructure for performing distributed collaborative computations while ensuring data security and privacy is properly protected. Our solution enables all parties to specify policies regulating access and processing of their data from others. In particular, the proposed solution supports three levels of visibility over data, which can be used by the parties to specify access restrictions to their datasets by granting a subject no visibility, full visibility, or partial visibility (i.e., encrypted visibility) over portions of the data. Since during a distributed computation data move from one network component to the other, our solution also captures the information content carried in the computation to ensure no information is improperly leaked.

The different steps (operations) of a computation can see the selective involvement of different parties as deemed desirable for economic or performance reasons. The proposed solution finds an assignment of operations to subjects, fully respecting the access restrictions and such that the computed assignment entails minimum economic cost for the execution of the computation. The idea consists in producing an extended version of the original computation, where the operations can be assigned to external subjects when economically convenient, and where the access restrictions are enforced by dynamically encrypting or decrypting data, to adjust the visibility over a data item whenever data are elaborated by a subject not authorized to access them in plaintext form. The injection of encryption-on-the-fly increases the possibilities to use data in collaborative computations and, when done to the aim of minimizing the economic cost of the computation, can provide for significant economic savings. Such benefits come however at the price of increasing the complication of computation optimization, which has to consider access restrictions and economic costs. Our approach nicely fits MARSAL and can then help enabling collaborative computations involving different data centers and computational providers.

We have also started the investigation of a probabilistic scheme that protects the integrity of computations

and that aims at providing a balance between the integrity guarantees offered and their overhead.

1.2 State of the art and MARSAL innovation

Previous work supporting secure collaborative computations falls in the area of distributed query processing (e.g., [26, 28]) and big data analytics (e.g., [2, 4, 32]). The main focus of this previous work is on the problem of obtaining efficient query plans (e.g., approaches that consider multiple cost metrics and parameters in optimizing the plans [36] or exploit multithreaded software on multicore hardware [23]), but do not take into consideration access restrictions. Access restrictions are supported in the relational database context through: views that provide access to only certain portions of the underlying relations (e.g., [12, 24, 33]); sovereign joins that rely on a secure coprocessor to perform queries over data stored in encrypted form [1]; access patterns that support the access to specific data sources only providing the values of certain attributes as input (e.g., [3, 6]); or data masking that ensure the proper protection of data returned by a query according to a privacy policy associated with the underlying relations (e.g., [27]). The main differences between these proposals and the work presented in the deliverable is that they do not consider encryption in the specification of restrictions.

Work closest to those presented in the following chapters has addressed the problem of protecting data confidentiality in distributed computations (e.g., [15, 29, 34, 39]). In [39] the authors present an approach to collaboratively execute queries on data that are subject to access restrictions defined by their owners. A query must be executed according to a query plan that entails only authorized data flows, while minimizing transfer costs. In particular, since different join evaluation strategies imply different information flows among the parties, the work in [39] defines multiple join evaluation strategies that can be interchangeably used to enforce access restrictions. In [34] the authors propose a privacy-aware operator placement approach (i.e., a way for finding the best subject in charge of executing an operator in a distributed system by satisfying privacy constraints and, at the same time, maximizing performance in query evaluation). The proposed solution is based on programming language techniques for regulating and controlling information flows. The proposal in [29] aims at protecting computations in hybrid clouds, preventing flows of sensitive information to the public cloud. In [15] the authors provide a solution for restricting access and sharing of distributed data, which supports the explicit consideration of join paths in the authorizations.

These works confirm the relevance of the problem, but focus on different aspects. In fact, the approach in [39] considers only data explicitly exchanged among providers and do not take into consideration implicit information disclosure. The approach in [15] presents a more expressive authorization model but, however, requires collaborative specification of authorizations. None of the proposals above-discussed consider the possibility of protecting data with encryption.

In [21, 22] the authors address a complementary problem that aims at 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 [13]. The approach in [20] integrates this authorization model in a distributed query optimizer. In this deliverable, we have introduced a novel approach for identifying an assignment of operations to subjects to minimize the economic cost associated with the execution of such operations, also taking into consideration the cost of encryption and decryption operations. Other related work complementary to ours has investigated the use of Trusted Execution Environments (e.g., Intel SGX) for storing or processing sensitive data in an otherwise non fully trusted scenario or host (e.g., [31, 35, 38]). The solution presented in the deliverable can be extended by considering the trusted execution environments to delegate part of a computation.

Several works (e.g., [25, 30, 37]) have also investigated the use and support of encryption for the protec-

tion of data in storage or query execution. Other solutions (e.g., [5, 7]) adopt 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., [16]) have designed techniques to verify the integrity of query results computed by potentially untrusted providers. All these solutions are complementary to our proposal.

1.3 Deliverable overview and report structure

The document is structured in three chapters.

- Chapter 2 “Policy specifications”: it provides a description of the policy-driven model that allows different parties of a distributed system to specify policies that protect the confidentiality of their data by regulating their release.
- Chapter 3 “Operations assignment”: it presents an approach that finds an allocation of the different operations of a computation to subjects to minimize a parameter of interest (e.g., cost or performance). Since we expect the economic cost to be the driving factor in the choice of the assignment of operations to subject, our solution finds an assignment of operations to subjects that takes into consideration the cost of operation execution as well as of the encryption/decryption operations needed to make the assignment authorized.
- Chapter 4 “Conclusions”: it states the findings of this document and draws conclusions for the ongoing and future development of the work in the project.

2. Policy specifications

We present an approach for supporting collaborative computations over data stored at different network components. In particular, we consider a Distributed Cloud Storage (DCS) infrastructure, where the storage nodes are under the control of different *infrastructure owners*. In addition to the storage service, the infrastructure owners can also offer computational services, with high elasticity and support for efficient execution of computationally intensive applications. The owner has authority over the data managed by its nodes and hence can regulate how they can be accessed by or shared with others. Since the information stored at the DCS can be involved in analysis, one of the objective of MARSAL is to facilitate distributed computations (e.g., applying data analytics to identify tourist flows from mobility datasets) while offering security and privacy guarantees.

Figure 2.1 illustrates this scenario, where multiple data centers (DCs) may need to collaborate for the evaluation of computations over their data. Note that the DCs can be placed at the different layers of the MARSAL architecture, e.g. two at the regional edge and one at the radio-edge of the MARSAL architecture. Computation can be realized possibly involving also external centers (i.e., centers not storing the data on which the computation is performed) that offer computing power at limited prices but that are not necessarily fully trusted to access the data content. The release of data for collaborative computations then needs to be regulated according to the infrastructure owners' requirements, making data release selective.

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 infrastructure owners. Execution of queries is performed according to a query plan that is represented as a tree $\mathtt{T(N)}$, with $\mathtt{N}$ 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. In the remainder of this document, to describe the working of the proposed approach, we will refer to the following running example.

Example 2.1 (Running example) *We consider two infrastructure owners* $\mathbb{A}$ *and* $\mathbb{C}$*, a subject* $\mathbb{S}$ *interested in performing an analysis involving the data managed by* $\mathbb{A}$ *and* $\mathbb{C}$*, and three external subjects* $\mathbb{X}$*,* $\mathbb{Y}$*, and* $\mathbb{Z}$ *offering computational resources only. Owner* $\mathbb{A}$ *manages the information about the access points of a cell-free network. Such information is stored in relation Access Point (*$\mathtt{AP}$*) with attributes* (APid, $\mathrm{NumberAntennas}$, $\mathrm{PLatitude}$, $\mathrm{MLongitude}$) *reporting information about the identifier of an access point (*A*), the number of antennas of the access point (*N*), and the coordinates (latitude and longitude, respectively) of the access point (*P*, and* M*, respectively). Infrastructure owner* $\mathbb{C}$ *manages the information about the activities of user devices. Such information is stored in relation* $\mathtt{CDR}$(Cid, IdAP, Hashid, Time) *reporting the information about the identifier of the relation (*C*), the identifier of an access point (*I*), the hash of the identification number of a user device (*H*), and the activity time (*T*). Figure 2.2(a) shows the schema of these two relations. Subject* $\mathbb{S}$ *is interested in retrieving the access point identifier and number of devices, for all the access points that have more than one antenna and that have been used by more than 100 devices in timeframe [t1,t2]. This analysis corresponds to the execution of the query shown in Figure 2.2(b) whose query plan is shown in Figure 2.2(c). In the query, t1 and t2 correspond to two timestamps (e.g., '09:00:00' and '17:00:00', respectively). Note that in the figure, and in the following, attributes are denoted using their initials and, for the sake of readability, in the authorizations*

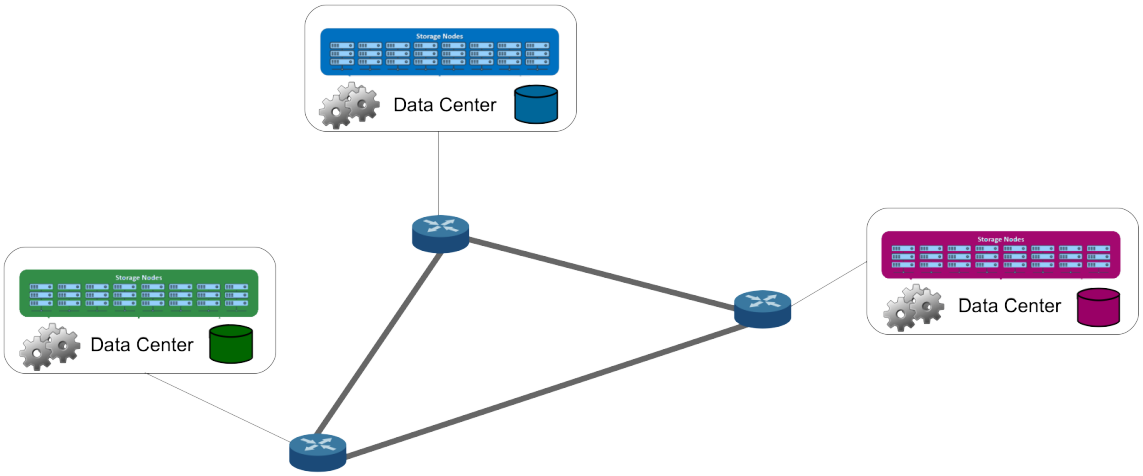


Figure 2.1: Storage and computation scenario

we denote a set of attributes simply with the sequence of attributes composing it, omitting the curly brackets and commas (e.g., CIHT *stands for* $\{\mathrm{C,I,H,T}\}$*).*

We now illustrate how the infrastructure owners can specify their access restrictions (authorizations) and how they can be enforced.

2.1 Authorizations and relation profiles

We now describe the format of the authorizations that each infrastructure owner can define (Section 2.1.1), and then we describe how to capture the informative content of a relation (Section 2.1.2).

2.1.1 Authorizations

Each infrastructure owner can specify access restrictions at the fine grain of attribute that correspond to three different levels of visibility of the attributes:

- *plaintext visibility*, meaning that the attribute's values are completely visible;
- *encrypted visibility*, meaning that the attribute's values are visible only in encrypted form;
- *no visibility*, meaning that the attribute's values are not visible (neither in plaintext nor in encrypted form).

Plaintext visibility and no visibility are the classical ways used to regulate the access to some information. Encrypted visibility is instead a peculiar characteristic of the model and supports the execution of operations over attributes without however granting the access to the actual values of the attributes themselves. In this way, subjects that are not completely trusted to access the data content can be involved in the execution of operations. Note that 'no visibility' does not need to be explicitly defined since we assume a *closed policy* for the specification of authorizations. This implies that only accesses to data for which there exists an authorization specifying plaintext or encrypted visibility are permitted; all the other accesses are denied by default.

AP

APid	**N**umAntennas	**P**Latitude	**M**Longitude

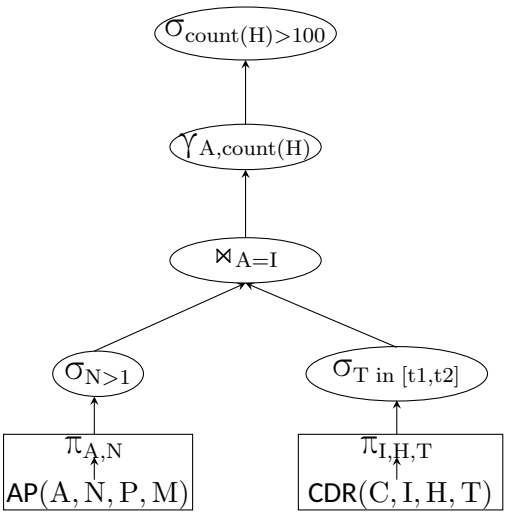
CDR

Cid	**I**dAP	**H**ashid	**T**ime

(a) data sources

```
SELECT A, COUNT(DISTINCT H)
FROM AP JOIN CDR ON A=I
WHERE N > 1 AND T BETWEEN t1 AND t2
GROUP BY A
HAVING COUNT(DISTINCT H)>100
```

(b) query



(c) query plan

[ANPM,_]→𝔸
[CIT,H]→𝔸

[AN,PM]→𝕄
[CIHT,_]→𝕄

[AN,PM]→ℂ
[CIHT,_]→ℂ

[A,NPM]→ℤ
[CH,IT]→ℤ

[AN,_]→𝕊
[CIHT,_]→𝕊

[AN,_]→any
[CI_]→any

[_,ANPM]→𝕏
[_,CIHT]→𝕏

(d) authorizations

Figure 2.2: An example of data sources (a), query (b), corresponding query plan (c), and of authorizations on relations AP and CDR (d)

Formally, given a relation R and a subject S, an authorization over R for S is a rule of the form [R_1,R_2]→S. with $R_1, R_2 \subseteq R$ and $R_1 \cap R_2 = \emptyset$. This rule states that S can access in plaintext the set R_1 of attributes, and in encrypted form the set R_2 of attributes. Note that the plaintext visibility over an attribute a of relation R also implies the encrypted visibility over the same attribute since the encrypted representation of attribute values conveys less information than the corresponding plaintext values. Furthermore, since the subjects that can be involved in the execution of a query and for which an infrastructure owner has to define an authorization may not be known a priori, the proposed approach supports the specification of authorizations that apply to all subjects when no explicit authorization already exists. Such authorizations have "any" as subject of the authorization.

Example 2.2 (Authorizations) *Figure 2.2(d) illustrates the authorizations defined for our running example. Each infrastructure owner is authorized to access all the attributes in its relation in plaintext (e.g., $\mathbb{A}$ can access in plaintext all the attributes of relation* AP *and $\mathbb{C}$ can access in plaintext all the attributes of relation* CDR*), and possibly also attributes of other relations in plaintext or encrypted (e.g., $\mathbb{A}$ can access attribute H of relation* CDR *in encrypted form and in plaintext attributes CIT). External subjects offering computational resources only can access a subset of the attributes of the relations managed by the infrastructure owners in plaintext or encrypted (e.g., $\mathbb{X}$ can access all attributes of relations* AP *and* CDR *in encrypted form, and $\mathbb{Z}$ can access attribute A of relation* AP *in plaintext and attributes NPM of relation* AP *in encrypted form). For all subjects that are not known a priori, there are two authorizations with value 'any' as subject stating that attributes AN of relation* AP *and attributes CI of relation* CDR *can be accessed in plaintext form.*

2.1.2 Relation profile

To verify whether a subject can access a relation (base or derived from the evaluation of a query), it is necessary to capture its informative content. To this purpose, each relation is associated with a *relation profile* that describes the schema of the relation, that is, the information explicitly visible in the relation, as well as the implicit information that is not visible in the relation but that has left a trace in the relation itself. In particular, for this implicit information we distinguish between *implicit attributes*, *equivalent attributes*, and *renamed attributes*.

Implicit attributes are those attributes used in the computation of the relation. For instance, for the query in Figure 2.2(b), the selection condition 'T BETWEEN t1 AND t2' leaks the fact that the result of the query has been obtained considering all the tuples that have value of T in the range [t1,t2].

Equivalent attributes are sets of attributes that have been connected in the computation of the relation. For instance, for the query in Figure 2.2(b), condition A=I implies precise leakage of the values of A from the visibility of I (and vice versa).

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 A as W from AP" reveals the values of attribute A under attribute name W, but no authorization regulates the release of W. Clearly, the authorizations originally defined over A must apply also to W, since W is just a different name for A. We refer to W as the renamed version of A. The concept of renamed attribute applies to both explicit as well as implicit attributes, and to plaintext as well as encrypted attributes.

Since both the attributes visible in a relation as well as the implicit attributes can be in plaintext or encrypted form, a relation profile is defined as a 6-tuple $[R^{vp}, R^{ve}, R^{ip}, R^{ie}, R^{\simeq}, R^{\lhd}]$ 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'. Graphically, the relation profile of a relation obtained through the execution of the operation represented by a node is shown as a dotted flag attached to the node itself 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.

The profile of a base relation has all the elements but R^{vp} empty, since it is assumed accessible in plaintext (which, however, does not imply that it is stored in plaintext but only that it is accessible in plaintext by the subject storing it), and does not carry any implicit content or equivalence/renaming relationship. 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. Figure 2.3 illustrates the graphical representation of the profiles resulting from relational operations, user defined function, encryption operation, and decryption operation. For the sake of readability, in the figure and in the following examples, 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).** The profile of the resulting relation contains, in the visible attributes, only those attributes that have been projected. All the others components of the profile are the same as the ones of the operand.

- **Selection (σ).** The profile of the resulting relation has the same visible and renamed attributes as the operand. 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 the visibility of a in the operand). For conditions of the form 'a_i *op* a_j', equivalence $\{a_i, a_j\}$ is added to the equivalence set.

- **Cartesian product ($\times$).** The profile of the resulting relation is obtained by taking the union of the corresponding sets in the profiles of the operands.

- **Join ($\bowtie$).** It is equivalent to a selection with a condition that is a Boolean formula of basic conditions of the form 'a_i *op* a_j', which is applied to the cartesian product of the operands (i.e., $\sigma_C(R_l \times R_r)$). The profile of the result reflects then the information conveyed by both these operators.

- **Group by (γ).** 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).

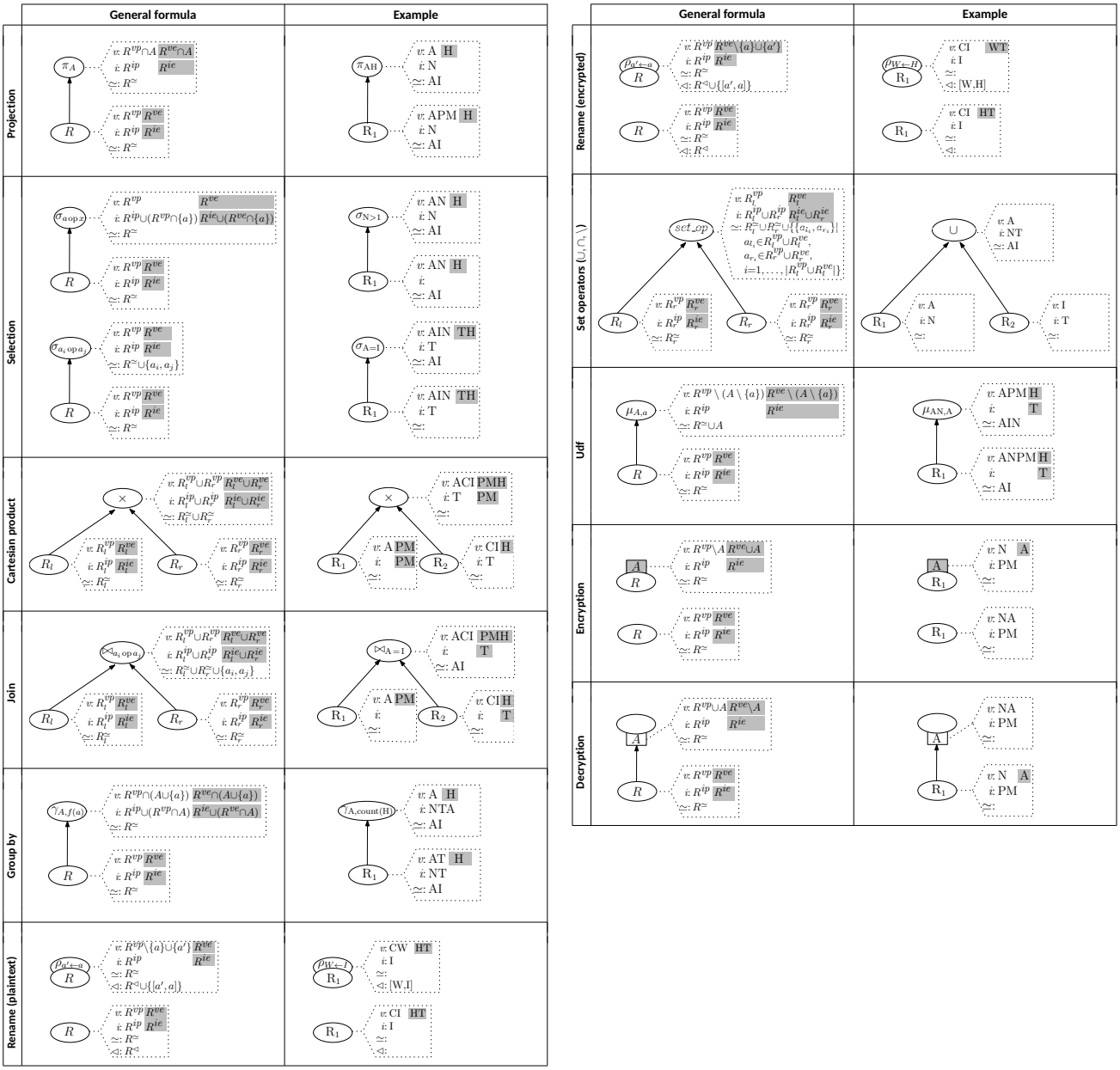


Figure 2.3: Graphical representation of the profiles resulting from relational, udf, encryption, and decryption operations

- **Rename (ρ).** 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. Function ω returns the original names of the attributes on which it applies (if any) or the attributes themselves (i.e., 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). The use of such a function 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.

- **Set operators ($\cup, \cap, \setminus$).** 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 operands' profiles. The fact that the i-th attribute a_{l_i} in the resulting relation derives 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 $\{a_{l_i}, a_{r_i}\}$ 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.** 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.** 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.

Example 2.3 (Relation profile) *Figure 2.4(a) illustrates the profiles of the relations resulting from the operations of our running example. Each node has, on its left, a set of subjects (we will elaborate on this in the next section). Also, the query plan does not include any encryption/decryption operation since they do not appear in the original query plan; we will illustrate how and why the query plan is extended with them in the next section.*

Note that we consider profiles where attributes have their original names, meaning that in case of renaming operations, new names of attributes are replaced with the original ones. 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 $R^{\triangleleft}$. To maintain the notation simple, we simply (and equivalently) assume profiles to be closed against the renaming relationship so to refer to attributes in the base relations [14]. In the following, we will then use the term profile to refer to the closed profile of a relation, and notation $[R^{vp}, R^{ve}, R^{ip}, R^{ie}, R^{\simeq}]$ to denote the (closed) profile of R.

It is important to note that in a query plan *i)* attributes appearing in the profile of the relation resulting from an operation will survive in the profiles of relations resulting from subsequent operations (i.e., they can move from one component to another, but they never disappear from the profile), and *ii)* equivalence sets can only increase going up in the query plan, meaning that when an attribute is inserted into an equivalence set, it will never be removed from it. These two observations are formally proven by the following theorem.

Theorem 2.1 *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) $(R_y^{vp} \cup R_y^{ve} \cup R_y^{ip} \cup R_y^{ie} \cup \{A \mid A \in R_y^{\simeq}\}) \subseteq (R_x^{vp} \cup R_x^{ve} \cup R_x^{ip} \cup R_x^{ie} \cup \{A \mid A \in R_x^{\simeq}\})$

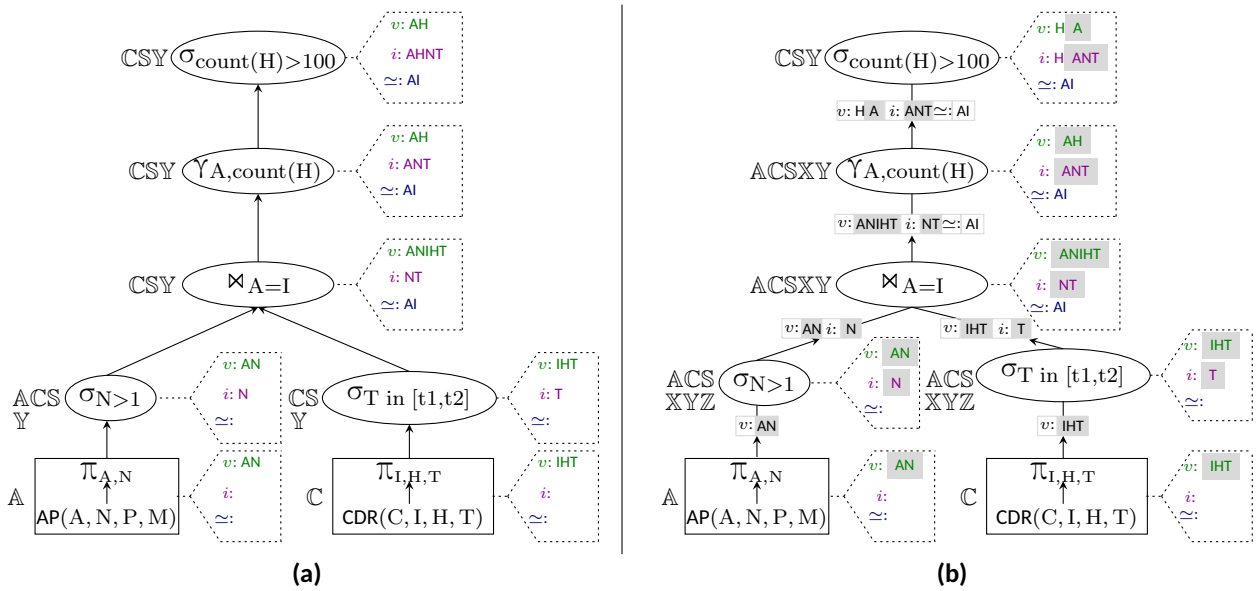


Figure 2.4: Query plan with relation profiles and candidates for the query in Figure 2.2(c) assuming all attributes in plaintext (a) and assuming minimum required view (b)

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 $\exists a \in \{R_y^{vp} \cup R_y^{ve} \cup R_y^{ip} \cup R_y^{ie} \cup \{A \mid A \in R_y^{\simeq}\}\}$ s.t. $a \notin \{R_x^{vp} \cup R_x^{ve} \cup R_x^{ip} \cup R_x^{ie} \cup \{A \mid A \in R_x^{\simeq}\}\}$. 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 Figure 2.3, 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 $R_x^{ip} \cup R_x^{ie}$. 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 2.3) are: selection (a would be in R_x^{ip}, R_x^{ie}, or $R_x^{\simeq}$); join (a would be in $R^{\simeq}$); group-by (a would be in R_x^{ip} or R_x^{ie}); a set operator (a would be in $R^{\simeq}$); and udf (a would be in $R^{\simeq}$). 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, $(R_y^{vp} \cup R_y^{ve} \cup R_y^{ip} \cup R_y^{ie} \cup \{A \mid A \in R_y^{\simeq}\}) \subseteq (R_x^{vp} \cup$

$R_x^{ve} \cup R_x^{ip} \cup R_x^{ie} \cup \{A \mid A \in R_x^{\simeq}\})$, 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.

1) Cartesian product (i.e., 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.

2) Selection/join (i.e., 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 vice versa); 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.

3) Set operator (i.e., 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 $A_i \in R_x^{\simeq}$ and $A \subset A_{yizi}$. This contradicts our hypothesis.

4) Udf (i.e., 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 $R_x^{\simeq}$ in place of A_i. Otherwise, it creates set A_x and inserts it into $R_x^{\simeq}$. Therefore, if $A_x \cap A = \emptyset$, then $A \in R_x^{\simeq}$. Otherwise, $A \subset A_{ix}$ and therefore $A \in R_x^{\simeq}$. This contradicts our hypothesis.

Note that rename operator does not have impact on the second conditions of the theorem, since renamed attributes are substituted by the corresponding original attribute names when the profile is closed.

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.

Since the two conditions hold, the theorem is proved. □

2.2 Authorizations enforcement

The concept of relation profile allows us to capture the informative content carried by a relation, and therefore 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 owners) might be involved in the control for the release of a derived relation obtained during the execution of the operations characterizing a query. In the following, for the sake of simplicity, we use notation $\mathcal{P}_S$ ($\mathcal{E}_S$, resp.) as a short-hand for the set of attributes that S is

	Authorizations		Authorized attributes	
Subject	AP(A,N,P,M)	CDR(C, I, H, T)	Plaintext	Encrypted
$\mathbb{A}$	[ANPM,_]→$\mathbb{A}$	[CIT,H]→$\mathbb{A}$	$\mathcal{P}_{\mathbb{A}}$=ANPMCIT	$\mathcal{E}_{\mathbb{A}}$=H
$\mathbb{C}$	[AN,PM]→$\mathbb{C}$	[CIHT,_]→$\mathbb{C}$	$\mathcal{P}_{\mathbb{C}}$=ANCIHT	$\mathcal{E}_{\mathbb{C}}$=PM
$\mathbb{S}$	[AN,_]→$\mathbb{S}$	[CIHT,_]→$\mathbb{S}$	$\mathcal{P}_{\mathbb{S}}$=ANCIHT	$\mathcal{E}_{\mathbb{S}}$=_
$\mathbb{X}$	[_,ANPM]→$\mathbb{X}$	[_,CIHT]→$\mathbb{X}$	$\mathcal{P}_{\mathbb{X}}$=_	$\mathcal{E}_{\mathbb{X}}$=ANPMCIHT
$\mathbb{Y}$	[AN,PM]→$\mathbb{Y}$	[CIHT,_]→$\mathbb{Y}$	$\mathcal{P}_{\mathbb{Y}}$=ANCIHT	$\mathcal{E}_{\mathbb{Y}}$=PM
$\mathbb{Z}$	[A,NPM]→$\mathbb{Z}$	[CH,IT]→$\mathbb{Z}$	$\mathcal{P}_{\mathbb{Z}}$= ACH	$\mathcal{E}_{\mathbb{Z}}$=NPMIT
any	[AN,_]→any	[CI,_]→any	$\mathcal{P}_{\text{any}}$=ANCI	$\mathcal{E}_{\text{any}}$=_

Figure 2.5: Authorizations and corresponding overall views for the subjects of our running example

authorized to access in plaintext (encrypted, resp.) form (i.e., $\mathcal{P}_S$ = {$a\in R_1 \mid [R_1, R_2]\rightarrow S$} and $\mathcal{E}_S$ = {$a\in R_2 \mid [R_1, R_2]\rightarrow S$}). Figure 2.5 shows the authorizations for our running example and the corresponding overall views for the different subjects.

Given a (base or derived) relation R with its profile $[R^{vp}, R^{ve}, R^{ip}, R^{ie}, R^{\simeq}]$, a subject is authorized to access the relation if all the following three conditions are satisfied.

1. The subject is authorized to access in plaintext all the attributes, visible and implicit, represented in plaintext (i.e., $R^{vp}\cup R^{ip} \subseteq \mathcal{P}_S$).
2. The subject is authorized to access in plaintext or in encrypted form all the attributes, visible and implicit, represented in encrypted form ($R^{ve}\cup R^{ie} \subseteq \mathcal{P}_S\cup\mathcal{E}_S$).
3. The subject is authorized to access in the same form (i.e., plaintext or encrypted) all the equivalent attributes, that is, all the attributes appearing in the same equivalence set in the relation ($\forall A\in R^{\simeq}$, $A\subseteq\mathcal{P}_S$ or $A\subseteq\mathcal{E}_S$).

Conditions 1 and 2 correspond to an enforcement of the authorizations considering both the visible and implicit attributes. Condition 3 prevents unintended information leakage of attribute values due to comparisons in query evaluation.

Example 2.4 (Visibility of a relation) *Consider the authorizations in Figure 2.5 and a relation R with profile:* [H,AIPM,_,_,{AI}]*:*

- $\mathbb{C}$ *is authorized for* R*;*
- $\mathbb{A}$ *is not authorized for R (condition 1, attribute* H*);*
- $\mathbb{S}$ *is not authorized for R (condition 2, attributes* PM*);*
- $\mathbb{Z}$ *is not authorized for R (condition 3, attributes* AI*).*

Another aspect involved in the enforcement of authorizations concerns regulating the assignment of operations in a query plan to a subject in respect of the authorization policy. 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. Intuitively, a subject can be considered authorized for the execution of an operation if and only if it is authorized for all the relations involved: the operand(s) of the operation, and the operation 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. Given a query plan $\mathtt{T(N)}$, our goal is to produce an authorized assignment of operations to subjects. The assignment of operations to subjects can be formally defined through a function $\lambda : \mathtt{N} \rightarrow \mathcal{S}$, with $\mathcal{S}$ a set of subjects. The function associates with each node n of a query plan $\mathtt{T(N)}$ the set $\lambda(n)$ of authorized assignees. An authorized assignment can be determined by extending the query plan with encryption and decryption operations to adjust visibility of attributes as required by the authorizations or by the operation requirements (i.e., when some operations cannot be evaluated over encrypted data). In particular, encryption can be used to protect 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 for the computation. We refer to a query plan enriched with encryption/decryption operations as an *extended query plan*. Given a query plan $\mathtt{T(N)}$, there may exist several extended query plans and, for each extended query plan, different assignments of subjects to operations that satisfy authorizations. In the following, the set of extended query plans for $\mathtt{T}$ is denoted $\mathcal{T}$. The encryption needed to make an assignment authorized depends on the actual subject to which an operation is assigned. For instance, for the query in Figure 2.2(c), attributes AI 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 $\mathbb{C}$. To determine an extended query plan, we then apply an approach that consists in first determining the set of *candidates* for operations, and then in injecting encryption and decryption operations only as needed, according to the selected assignment of operations to subjects. In the following, given a query tree plan $\mathtt{T(N)}$, function $\Lambda : \mathtt{N} \rightarrow 2^{\mathcal{S}}$ associates with each $n \in \mathtt{N}$ the set of candidates for the execution of n.

Different strategies can be adopted for finding the set of candidates. Two opposite strategies that can be followed consist in *1)* leaving plaintext visibility of all attributes or *2)* encrypting all attributes. The first strategy has the advantage of having operations always performed over plaintext data. However, it limits the number of candidate subjects to which each operation can be assigned. Figure 2.4(a) illustrates for the query of the running example the candidates for each operation in the query when we apply the first strategy. As we can see, only subjects $\mathbb{C}$, $\mathbb{S}$, and $\mathbb{Y}$ can perform the selection over attribute T as well as all the following operations in the query.

The second strategy has the advantage of increasing the number of subjects to which each operation can be assigned but it would prevent the evaluation of operations requiring plaintext visibility over attributes. For instance, for the query of our running example we have assumed that the last selection cannot be performed over encrypted values. Encrypting attribute H would prevent the possibility of executing such a selection. To avoid this problem and to have the possibility of selecting an assignment considering a larger number of candidates, we associate each node n in the query plan producing relation R with its *minimum required views* (MRVs), denoted $\hat{R}$. Intuitively, the minimum required view of a node operand for the execution of an operation is the operand relation where all (visible) attributes are encrypted but those that need to be in plaintext for the execution of the operation. All subjects that are authorized for the minimum required view of the operand(s) can be considered candidates for executing the operation. It is interesting to note that the set of candidates along a query plan $\mathtt{T(N)}$ enjoys a monotonic behavior. Figure 2.4(b) illustrates (in boxes on the arcs from the operands to the operations) the profiles of the minimum required views for the 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 relation $\mathtt{AP}$ for the execution of the join has all attributes (AN) visible and encrypted. For each $n \in \mathtt{N}$, 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 at-

tributes 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 2.2 *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 non-leaf children, if any.* $\hat{R}_l^{vp} \cup \hat{R}_r^{vp} \subseteq \hat{R}^{ip} \implies \Lambda(n_x) \subseteq \Lambda(n), \forall n_x$ *ancestor of* n.

Proof: Let us first analyze the case in which n_x is the direct ancestor of n in $\mathtt{T(N)}$ and assume, by contradiction, that $\exists S \in \Lambda(n_x)$ such that $S \notin \Lambda(n)$. By definition of candidate, 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 $\hat{R}_r$, with $\hat{R}_l$ and $\hat{R}_r$ the operands of node n. By Theorem 2.1, 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 that she is authorized to access in encrypted form, which is represented in plaintext in R and in encrypted form in R_x. In other words, there must exist an attribute $a \in \mathcal{E}_S$ such that 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 2.3), 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 of minimum required view, 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 $\forall n, n_x$ such that n_x is the direct ancestor of n, $\hat{R}_l^{vp} \cup \hat{R}_r^{vp} \subseteq \hat{R}^{ip} \implies \Lambda(n_x) \subseteq \Lambda(n)$, for the transitivity of operator $\subseteq$, the theorem holds. □

Operations can be assigned to any candidate, since visibility over the attributes can always be adjusted by inserting on-the-fly encryption operations as demanded by authorizations, without affecting the evaluation of the operation, as stated by the following theorem.

Theorem 2.3 *Let* $\mathtt{T(N)}$ *be a query plan, and* $\Lambda : \mathtt{N} \to 2^S$ *be a candidate assignment function for it:*

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

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

Proof: For the sake of readability, to clearly distinguish between nodes of the original tree $\mathtt{T(N)}$ and the same nodes in the extended tree $\mathtt{T'(N)}$, we will denote with n' the counterpart in $\mathtt{T'(N)}$ of node n in $\mathtt{T(N)}$. We now separately prove the two conditions of the theorem.

i) Suppose, by contradiction, that $\exists S = \lambda(n')$ such that $S \notin \Lambda(n)$, 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 the following two scenarios.

- $\exists a$ in the profiles of n, $\hat{R}_l$, and/or $\hat{R}_r$ such that 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 2.1 states that all attributes in the profile of a relation also belong to the profile of its ancestor. Therefore, if a does not belong to the profile of n (n', resp.), then a certainly 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 $\hat{R}_r$. The computation of minimum required views does not change which attributes are included in the profile of a node. This implies that the set of attributes in the profile of n be the same of n', thus contradicting our hypothesis.

- $\exists a$ appearing plaintext in the profiles of n, $\hat{R}_l$, and/or $\hat{R}_r$ such that 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 of minimum required view, a is needed plaintext for the execution of the operation in n (as otherwise it would be encrypted in $\hat{R}_l$ and $\hat{R}_r$, and therefore also in the profile of the relation resulting from n). If, however, a were to be plaintext for the execution of the operation in n, 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 $\mathtt{T(N)}$, 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 $\mathtt{T'(N)}$, 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 $\nexists \mathtt{T'(N)}$ such that $\mathtt{T'(N)}$ is an extended plan for $\mathtt{T(N)}$ for which λ is an authorized assignment. This can occur in the following two scenarios.

- $\exists a$ in the profiles of n', n'_l, and/or n'_r such that 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 $\mathtt{T(N)}$ and of its counterpart in $\mathtt{T'(N)}$ include the same set of attributes.

- $\exists a$ appearing plaintext in the profiles of n', n'_l, and/or n'_r such that 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 $\hat{R}_l$ and $\hat{R}_r$ (and, therefore, also in the profile of the relation resulting form n), plaintext visibility over a is not required to execute the operation in n'. Then, $\mathtt{T'(N)}$ can be extended with an encryption operation over a preceding n'. Plaintext visibility could however be needed to evaluate n'_l or n'_r. In this case, a would however leave a trace in the implicit plaintext component of the node profile.
  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 $\mathtt{T}'(\mathtt{N})$. 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, $\mathtt{T}'(\mathtt{N})$ 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 Figure 2.3).

Since both conditions hold, the theorem is proved. □

2.3 Summary

This chapter has illustrated an approach that allows different infrastructure owners to define how their data should be accessed by others, and that correctly enforces such access restrictions during the collaborative execution of a computation. The following chapter will describe how to determine the most suitable candidate for each operation in the computation.

3. Operations assignment

Given a plan for the execution of a computation (query), we have to find an assignment of the different operations composing the plan such that the assignment is authorized (i.e., compliant with the restrictions that each infrastructure owner has defined over its data). According to the discussion in the previous chapter, for any operation in the query plan, only subjects in the operation's candidate set need to be considered (Theorem 2.3, *i*). Also, any of them would do, since any assignment taken from the candidate set can be made authorized by inserting encryption and decryption operations (Theorem 2.3, *ii*), resulting in different extended authorized query plans. Among these extended authorized query plans, we can select the one that optimizes a parameter of interest such as cost or performance. In particular, we consider the economic cost in the choice of the assignment of operations to candidates. Each authorized assignment may then result in different costs and require injection of different encryption/decryption operations.

Example 3.1 (Extended query plans) *Figure 3.1 illustrates two extended query plans for our running example (Example 2.1), assuming operations allocated to the subject indicated to the left-hand side of each node. In the plan in Figure 3.1(a) attribute* H *of relation* CDR *is encrypted before being transmitted to* $\mathbb{A}$ *since* $\mathbb{A}$ *cannot access it in plaintext. In the plan in Figure 3.1(b) all attributes of the two relations* AP *and* CDR *are encrypted before being transmitted to* $\mathbb{X}$ *since* $\mathbb{X}$ *cannot access them in plaintext. Note also that attributes* N *and* T *must be encrypted before the execution of the select operations because otherwise the operations would leave an implicit plaintext trace in the computation that* $\mathbb{X}$*, executing the subsequent operations, cannot access. In both plans, count(*H*) is decrypted before the execution of the final selection because we assume that the execution of this operation needs to access plaintext values.*

Note that encryption and decryption operations are assigned to the same subject of the nodes to which they are attached. Indeed, the subject authorized for the operation represented by 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 $\mathtt{T}$ and a cost function κ, we aim at identifying an assignment λ' and an extended authorized query plan $\mathtt{T}'$ for λ' that minimizes the economic cost $\kappa(\lambda',\mathtt{T}')$ of the evaluation of $\mathtt{T}'$ according to assignment λ'. Formally, our minimization problem can be formulated as follows.

Problem 3.1 (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* $\kappa{:}\Lambda \times \mathcal{T} \rightarrow \mathbb{R}$*, 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''*,* $\kappa(\lambda',\mathtt{T}') \leq \kappa(\lambda'',\mathtt{T}'')$*.*

The computation of a minimum cost assignment needs to consider the cost of executing a computation, the cost of transferring data between different subjects involved in the computation, and also possible encryption/decryption that would be required for the selected candidates to be authorized for the operations assigned to them.

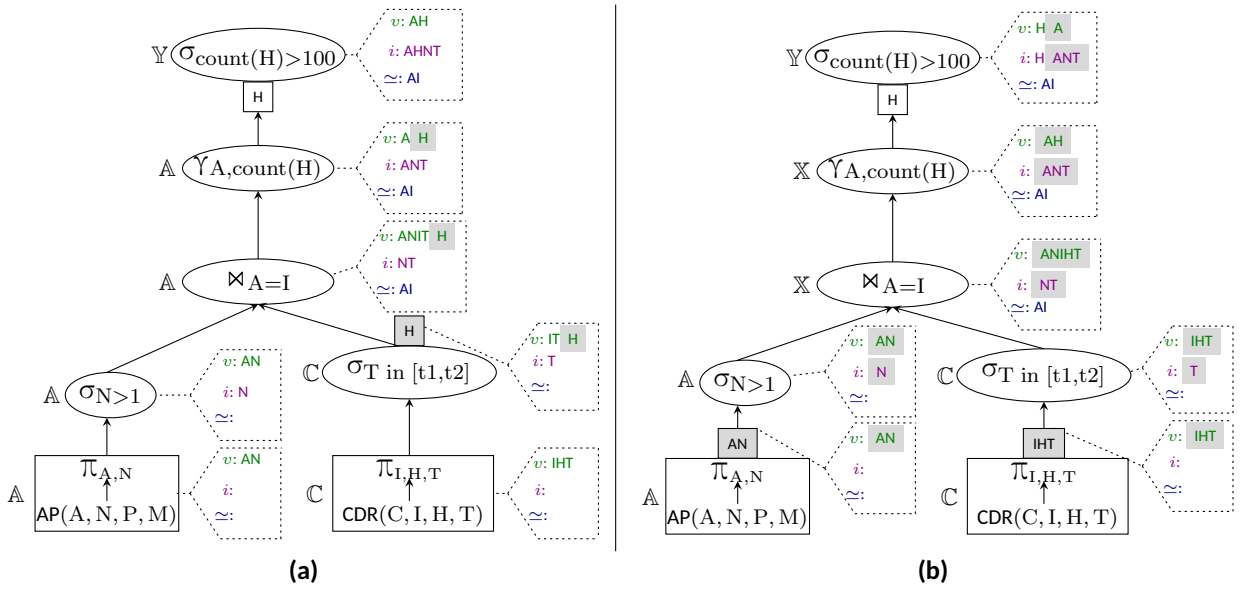


Figure 3.1: An example of two extended authorized query plans

3.1 Computing a minimum cost assignment

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

1. compute the candidate assignment function Λ for $\mathtt{T(N)}$, that is, identify the set of candidate subjects for the execution of each node in $\mathtt{N}$;

2. determine the assignment λ in Λ (i.e., such that $\forall n \in \mathtt{N}: \lambda(n) \in \Lambda(n)$) such that the extended query plan $\mathtt{T}'$ 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 trees (Theorem 2.2). 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 3.1, step 2 above) can be translated into a binary programming problem by illustrating the corresponding variables, constraints, and objective function.

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

Figure 3.2: Variables of the binary programming problem

3.1.1 Variables

Figure 3.2 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 variables. They model the solution of our minimization problem. In particular, given a query plan $\mathtt{T(N)}$, the output variables model an authorized assignment function λ for $\mathtt{T(N)}$, and the format (plaintext or encrypted) of the, visible or implicit, attributes in the relation profile associated with each node $n \in \mathtt{N}$.

- *Assignments.* For each subject $s \in \mathcal{S}$ and each node $n \in \mathtt{N}$, 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 $n \in \mathtt{N}$, 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 or 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 view (needed to guarantee the correctness of node profiles in terms of the completeness 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 2.4(b)), 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 minimum required view over n for the execution of the parent node $\bar{n}$.

3.1.2 Constraints

Constraints restrict combination of values for the output variables described in Section 3.1.1 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.*

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

  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.*

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

  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, 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.

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

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

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

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

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 2.1.

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

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

  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).

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

  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.

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

  (3.10) $\forall n{\in}\{n{\in}\mathtt{N}{:}\pi_{n,\bar{n}}{=}0, \forall \bar{n}{\in}\mathtt{N}\}$: $vp_{a,n} + ve_{a,n}{=}0, \forall a{\notin}R$

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

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.

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

  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.

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

  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.

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

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

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 3.7 and 3.8). 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.

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

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

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

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

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).

3.1.3 Objective function

The objective function of the binary programming problem models the economic cost of the evaluation of the extended authorized query plan $\mathtt{T}'(\mathtt{N})$ 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:

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

We now describe each of the three cost components. Figure 3.3 summarizes the cost parameters that will be used in such cost components.

$cost_s$:	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

Figure 3.3: Cost parameters

Operation execution. The cost of executing the operations in the query plan is the sum of the costs of executing the different node's 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:

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

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:

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

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:

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

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 subject requesting query execution (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 subject who submitted the query. This extra node permits to keep into consideration the cost of transferring the query result to the subject requesting query execution as well as the cost of the decryption operations performed by such a subject on the encrypted attributes in the query result.

Figure 3.4 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 3.2, 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 3.1.

3.2 Computing and distributing assignments

We now 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 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 indeed equivalent). To

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Figure 3.4: Binary programming for computing a minimum cost assignment

this purpose, 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 3.1. The pair of attributes A and I must be encrypted with the same key since they belong to the equivalence set of the profile of the root node. The keys to be established for a query plan execution is determined considering the set of attributes involved in encryption operations, and the fact that attributes appearing together in an equivalence set must be encrypted with the same key. The key associated with an attribute (or

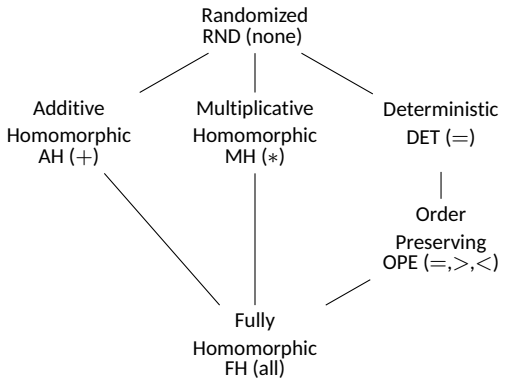


Figure 3.5: Functionality of encryption schemes

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 3.1(a), attribute H is subject to an encryption operation. Key k_{H} is distributed to $\mathbb{C}$. For the query plan in Figure 3.1(b), all attributes A, N, I, H, and T are subject to an encryption operation and the pair A,I must be encrypted with the same key. The keys involved in the execution are therefore k_{AI} distributed to $\mathbb{A}$ and $\mathbb{C}$, k_{N} distributed to $\mathbb{A}$, k_{H}, and k_{T} distributed to $\mathbb{C}$.

Encryption algorithm. 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 [17, 37]. As a matter of fact, the ability to operate on encrypted data (Figure 3.5) comes with some 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 relationships 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 subject requesting query execution 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

S	Receives (req_S)	Performs (q_S)
$\mathbb{Y}$	$[[q_\mathbb{Y}$,(H,k_H)]$_{\text{pri}_\mathbb{S}}]_{\text{pub}_\mathbb{Y}}$	SELECT A,decrypt(H^k,k_H) as H FROM $[\![\text{req}_\mathbb{A}]\!]$ WHERE H >100
$\mathbb{A}$	$[[q_\mathbb{A}$,-]$_{\text{pri}_\mathbb{S}}]_{\text{pub}_\mathbb{A}}$	SELECT A,COUNT(DISTINCT H^k) as H^k FROM AP JOIN $[\![\text{req}_\mathbb{C}]\!]$ ON A=I WHERE N > 1 GROUP BY A
$\mathbb{C}$	$[[q_\mathbb{C}$,(H,k_H)]$_{\text{pri}_\mathbb{S}}]_{\text{pub}_\mathbb{C}}$	SELECT I, T, encrypt(H,k_H) FROM CDR WHERE T BETWEEN t1 AND t2

Figure 3.6: Query dispatch for the plan in Figure 3.1(a)

potentially be known (of course with conditions operating on encrypted values when demanded by encryption operations in the plan). Figure 3.6 illustrates the query dispatch for the plan in Figure 3.1(a). In the figure, term $\text{req}_\mathbb{S}$=$[[q_S,(A_1,k_1),\ldots,(A_n,k_n)]_{\text{pri}_\mathbb{S}}]_{\text{pub}_S}$ represents the request (signed with $\text{pri}_\mathbb{S}$ 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 $\mathbb{S}$ to $\mathbb{Y}$ ($\text{req}_\mathbb{Y}$), which will call the sub-query at $\mathbb{A}$ ($\text{req}_\mathbb{A}$), which in turn will call the sub-query at $\mathbb{C}$ ($\text{req}_\mathbb{C}$).

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 owners, the query optimizer has to be trusted for such an enforcement. Each owner will perform a control at its side, before releasing the data to a third subject, to check that the subject requesting query execution is authorized for the released data. In fact, a subject requesting query execution must be authorized to access all data that are input to the query, which correspond to the base relations. The subject requesting the execution of a query is then trusted to involve other authorized subjects. With respect to the authorization enforcement, for simplicity, we have assumed the control of the authorizations holding for a given subject S as a check against the set $\mathcal{P}_S$ ($\mathcal{E}_S$, respectively) summarizing the attributes that S is authorized to access plaintext (in encrypted form, respectively). While the realization of such a control can be possible, in real applications we can expect authorizations over the different relations to be stored in a distributed manner (like the relations) and remain under the control of the respective owners. 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 authorizations. For instance, an owner can: *i)* publish its access control policy (which would then be 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.

3.3 Summary

This chapter has described an approach to determine an assignment of operations to subjects so to minimize the economic cost of collaborative computation evaluation, taking into consideration the encryption/decryption, operation execution, and transfer costs.

4. Conclusions

This deliverable has described the progresses achieved in the development of advanced security solutions within MARSAL. In particular, the deliverable has focused on the work done in task T5.2 on “Policy-driven data integrity and privacy protocols in multi-tenant infrastructures”.

The deliverable has presented an approach developed in MARSAL for enabling the protection of data in distributed computations, allowing collaborative computations in obedience of security policies that restrict access to the involved data and such that the cost associated with the execution of the operations in the computation is minimized. In particular, Chapter 2 has presented the authorizations that each infrastructure owner can define over its data. Such authorizations support three levels of visibility over data (no visibility, full visibility, and partial visibility over encrypted versions of data). Chapter 2 has then described how to enforce authorizations. Chapter 3 has then formalized the problem of computing an assignment of the operations in a computation to subjects such that the economic cost associated with the execution of the computation is minimized. Chapter 3 has also discussed an optimization model for modeling and solving the formulated problem through its interpretation as a binary programming problem.

Future work includes the implementation of the solution illustrated in this deliverable for computing operation assignments in obedience of authorizations while minimizing execution cost. Work will also continue the investigation on probabilistic solutions for assessing integrity of computations distributed to different nodes in the system. The aim is the definition of a model capturing the main characteristics of probabilistic techniques, namely the use of pre-computed and replicated tasks, so to enable their controlled generation and injection in such a way that provide best effective in achieving integrity guarantees. This work will be reported in deliverable D5.4.

References

[1] R. Agrawal, D. Asonov, M. Kantarcioglu, and Y. Li. Sovereign joins. In *Proc. of ICDE*, Atlanta, GA, USA, April 2006.

[2] W. Alkowaileet, S. Alsubaiee, M.J. Carey, C. Li, H. Ramampiaro, P. Sinthong, and X. Wang. End-to-end machine learning with Apache AsterixDB. In *Proc. of DEEM*, Houston, TX, USA, 2018.

[3] A. Amarilli and M. Benedikt. When can we answer queries using result-bounded data interfaces? In *Proc. of PODS*, pages 281–293, Houston, TX, USA, 2018.

[4] M. Armbrust, R.S. Xin, C. Lian, Y. Huai, D. Liu, J.K Bradley, X. Meng, T. Kaftan, M.J. Franklin, A. Ghodsi, and M. Zaharia. Spark SQL: Relational data processing in Spark. In *Proc. of SIGMOD*, pages 1383–1394, Melbourne, Australia, May-June 2015.

[5] J. Bater, G. Elliott, C. Eggen, S. Goel, A. Kho, and J. Duggan. SMCQL: Secure query processing for private data networks. *PVLDB*, 10(6):673–684, 2017.

[6] M. Benedikt, J. Leblay, and E. Tsamoura. Querying with access patterns and integrity constraints. *PVLDB*, 8(6):690–701, 2015.

[7] S. Chow, J-H Lee, and L. Subramanian. Two-party computation model for privacy-preserving queries over distributed databases. In *Proc. of NDSS*, San Diego, CA, February 2009.

[8] Jeffrey Cohen, Brian Dolan, Mark Dunlap, Joseph M. Hellerstein, and Caleb Welton. Mad skills: New analysis practices for big data. *PVLDB*, 2(2):1481–1492, 2009.

[9] S. De Capitani di Vimercati, D. Facchinetti, S. Foresti, G. Oldani, S. Paraboschi, M. Rossi, and P. Samarati. Artifact: Scalable distributed data anonymization. In *Proc. of the 19th International Conference on Pervasive Computing and Communications (PerCom 2021)*, Kassel, Germany (virtual), March 2021.

[10] S. De Capitani di Vimercati, D. Facchinetti, S. Foresti, G. Oldani, S. Paraboschi, M. Rossi, and P. Samarati. Multi-dimensional indexes for point and range queries on outsourced encrypted data. In *Proc. of the 2021 IEEE Global Communications Conference (GLOBECOM 2021)*, Madrid, Spain, December 2021.

[11] S. De Capitani di Vimercati, D. Facchinetti, S. Foresti, G. Oldani, S. Paraboschi, M. Rossi, and P. Samarati. Scalable distributed data anonymization. In *Proc. of the 19th International Conference on Pervasive Computing and Communications (PerCom 2021)*, Kassel, Germany (virtual), March 2021.

[12] S. De Capitani di Vimercati, S. Foresti, S. Jajodia, G. Livraga, S. Paraboschi, and P. Samarati. Fragmentation in presence of data dependencies. *IEEE TDSC*, 11(6):510–523, 2014.

[13] S. De Capitani di Vimercati, S. Foresti, S. Jajodia, G. Livraga, S. Paraboschi, and P. Samarati. An authorization model for multi-provider queries. *Proc. of the VLDB Endowment*, 11(3), November 2017.

[14] S. De Capitani di Vimercati, S. Foresti, S. Jajodia, G. Livraga, S. Paraboschi, and P. Samarati. An authorization model for query execution in the cloud. *The VLDB Journal*, 31(3):555–579, May 2022.

[15] S. De Capitani di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, and P. Samarati. Authorization enforcement in distributed query evaluation. *JCS*, 19(4):751–794, 2011.

[16] S. De Capitani di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, and P. Samarati. Efficient integrity checks for join queries in the cloud. *JCS*, 24(3):347–378, 2016.

[17] S. De Capitani di Vimercati, S. Foresti, G. Livraga, and P. Samarati. Practical techniques building on encryption for protecting and managing data in the cloud. In P. Ryan, D. Naccache, and J.-J. Quisquater, editors, *Festschrift for David Kahn*, pages 205–239. Springer, 2016.

[18] S. De Capitani di Vimercati, S. Foresti, G. Livraga, and P. Samarati. Digital infrastructure policies for data security and privacy in smart cities. In J. Vacca, editor, *Smart Cities Policies and Financing*. Elsevier, 2022.

[19] S. De Capitani di Vimercati, S. Foresti, G. Livraga, and P. Samarati. Towards owner-controlled data sharing. In P. Nicopolitidis, S. Misra, and L.T. Yang, editors, *Advances in Computing, Informatics, Networking and Cybersecurity: A Book Honoring Professor Mohammad S. Obaidat's Significant Scientific Contributions*. Springer, 2022.

[20] E.B. Dimitrova, P.K. Chrysanthis, and A.J. Lee. Authorization-aware optimization for multi-provider queries. In *Proc. of SAC*, pages 431–438, Limassol, Cyprus, 2019.

[21] N.L. Farnan, A.J. Lee, P.K. Chrysanthis, and T. Yu. Don't reveal my intension: Protecting user privacy using declarative preferences during distributed query processing. In *Proc. of ESORICS*, Leuven, Belgium, September 2011.

[22] N.L. Farnan, A.J. Lee, P.K. Chrysanthis, and T. Yu. PAQO: Preference-aware query optimization for decentralized database systems. In *Proc. of ICDE*, pages 424–435, Chicago, IL, USA, March–April 2014.

[23] J. Giceva, G. Alonso, T. Roscoe, and T. Harris. Deployment of query plans on multicores. *PVLDB*, 8(3):233–244, 2015.

[24] M. Guarnieri and D. Basin. Optimal security-aware query processing. *PVLDB*, 7(12):1307–1318, 2014.

[25] H. Hacigümüs, B. Iyer, S. Mehrotra, and C. Li. Executing SQL over encrypted data in the database-service-provider model. In *Proc. of SIGMOD*, pages 216–227, Madison, WI, USA, June 2002.

[26] D. Kossmann. The state of the art in distributed query processing. *ACM CSUR*, 32(4):422–469, 2000.

[27] M.M. Kwakye and K. Barker. Privacy-preservation in the integration and querying of multidimensional data models. In *Proc of PST*, pages 255–263, Auckland, New Zealand, December 2016.

[28] A.Y. Levy, D. Srivastava, and T. Kirk. Data model and query evaluation in global information systems. *JIIS*, 5(2):121–143, 1995.

[29] K.Y. Oktay, M. Kantarcioglu, and S. Mehrotra. Secure and efficient query processing over hybrid clouds. In *Proc. of ICDE*, pages 733–744, San Diego, CA, USA, April 2017.

[30] R.A. Popa, C. Redfield, N. Zeldovich, and H. Balakrishnan. CryptDB: Protecting confidentiality with encrypted query processing. In *Proc. of SOSP*, pages 85–100, Cascais, Portugal, October 2011.

[31] C. Priebe, K. Vaswani, and M. Costa. EnclaveDB: A secure database using SGX. In *Proc. of SP*, pages 264–278, San Francisco, CA, USA, March 2018.

[32] A. Rheinländer, U. Leser, and G. Graefe. Optimization of complex dataflows with user-defined functions. *ACM CSUR*, 50(3):38:1–38:39, 2017.

[33] S. Rizvi, A. Mendelzon, S. Sudarshan, and P. Roy. Extending query rewriting techniques for fine-grained access control. In *Proc. of SIGMOD*, pages 551–562, Paris, France, June 2004.

[34] G. Salvaneschi, M. Köhler, D. Sokolowski, P. Haller, S. Erdweg, and M. Mezini. Language-integrated privacy-aware distributed queries. *Proc. ACM Program. Lang.*, 3(OOPSLA), October 2019.

[35] C. Thoma, A.J. Lee, and A. Labrinidis. Behind enemy lines: Exploring trusted data stream processing on untrusted systems. In *Proc. of CODASPY*, pages 243–254, Richardson, TX, USA, March 2019.

[36] I. Trummer and C. Koch. Multi-objective parametric query optimization. *PVLDB*, 8(3):221–232, 2015.

[37] S. Tu, M.F. Kaashoek, S. Madden, and N. Zeldovich. Processing analytical queries over encrypted data. *PVLDB*, 6(5):289–300, 2013.

[38] D. Vinayagamurthy, A. Gribov, and S. Gorbunov. StealthDB: a scalable encrypted database with full SQL query support. *PoPETS*, 2019(3):370–388, 2019.

[39] Q. Zeng, M. Zhao, P. Liu, P. Yadav, S. Calo, and J. Lobo. Enforcement of autonomous authorizations in collaborative distributed query evaluation. *IEEE TKDE*, 27(4):979–992, 2015.