Enforcing Corporate Governance Controls with Cloud-based Services

Sabrina De Capitani di Vimercati *Senior Member, IEEE,* Sara Foresti *Senior Member, IEEE,* Stefano Paraboschi *Member, IEEE,* Pierangela Samarati *Fellow, IEEE*

Abstract—More and more organizations are today using the cloud for their business as a convenient alternative to in-house solutions for storing, processing, and managing data. Cloud-based solutions are then permeating almost all aspects of business organizations, resulting appealing also for sensitive or security critical applications, whose enforcement in the cloud requires however particular care. In this paper, we provide an approach for securely relying on cloud-based services for the enforcement of Internal Controls and Audit (ICA) functions for corporate governance. Our approach builds on a formalization of the ICA process and its requirements and on the consideration of the protection guarantees to be provided when outsourcing the process to external cloud services. The enforcement of the requirements leverages the use of selective encryption providing a self-protection layer on the data and on ICA reports, the hierarchical organization of keys based on the organizational structure, and compact tags for regulating write operations. Our solution enables the management of the ICA process with cloud-based services, while ensuring satisfaction of the protection requirements.

Index Terms—Cloud-based services; outsourcing; internal controls and audit process; access control; selective encryption

1 Introduction

Corporate Governance is a collection of rules, best practices, and processes needed to achieve an organization's objectives and strategies. The benefits and the importance of having a good corporate governance are continuously increasing, due to the growth in business regulation and capital mobility. Indeed, countries see corporate governance as a key factor for their global competitiveness. There are therefore several attempts to promote the adoption of national Corporate Governance Codes by organizations (e.g., the European Corporate Governance Codes Networks, www.ecgcn.org). An important role in these Corporate Governance Codes is given to the internal control and risk management system, which has, as central mechanisms, the *Internal Controls and Audit* (ICA) functions.

ICA functions aim mainly at verifying the effectiveness and efficiency of *operations*, and their compliance with internal rules, regulations, and laws. Their successful realization can contribute to the improvement of the quality of corporate governance and management. ICA functions can be performed in several ways, depending on how the general principles described in the Corporate Governance Code are implemented in the context of a specific organization. This paper focuses on the case of an organization structured in multiple units and where all the operations performed in the units must be checked according to a three-level ICA process. A structure with three levels is the most common in companies that have to comply with market regulations, like banks and financial institutions. The first level of control is executed by an employee of the unit where the operation has been performed. The second level of control is performed by the director of the unit. Finally, the third level of control is performed by an independent auditor. Each level of control aims at verifying different aspects related to, for example, the operational and business area, and produces a *report* summarizing the results of the control.

Due to the huge number of operations that are processed on a daily basis, the adoption of cloud-based services, with the involvement of external providers offering storage and access service functionality may result convenient for the management of the ICA process (see Figure 1). However, given the nature of the operations and their processing, it is critical to ensure their confidentiality, even against the external provider, which is not under the direct control of the organization. As a matter of fact, while the provider may be assumed to be trustworthy providing storage and access functionality, it may be not trusted to read the operations content and reports (*honest-but-curious* paradigm [2]). Also, the provider cannot be trusted by itself for the enforcement of the ICA process.

Although several proposals exist for the secure outsourcing of data and computations in the cloud, none of them can be directly applied to correctly enforce the ICA process of corporate governance. In this paper, we provide an approach for relying on an external cloud provider for the management of the ICA process, leveraging the provider not only for data storage and access services, but also for supporting the regulation and enforcement of ICA operations, while ensuring confidentiality of the data and processes on them against the provider itself. The contribution of the paper is multifold. First, we provide a characterization of the requirements for the ICA process and their formalization, capturing the rules to be enforced. Second, we provide an approach for external storage of operations and results of

- *S. De Capitani di Vimercati, S. Foresti, P. Samarati are with the Università degli Studi di Milano, Italy*
  Email: name.surname@*unimi.it*
- *S. Paraboschi is with the Università degli Studi di Bergamo, Italy*
  Email: parabosc@unibg.it

A preliminary version of this paper appeared under the title "Enforcing Corporate Governance's Internal Controls and Audit in the Cloud," in Proc. of IEEE CLOUD, Beijing, China, October 2020 [1].

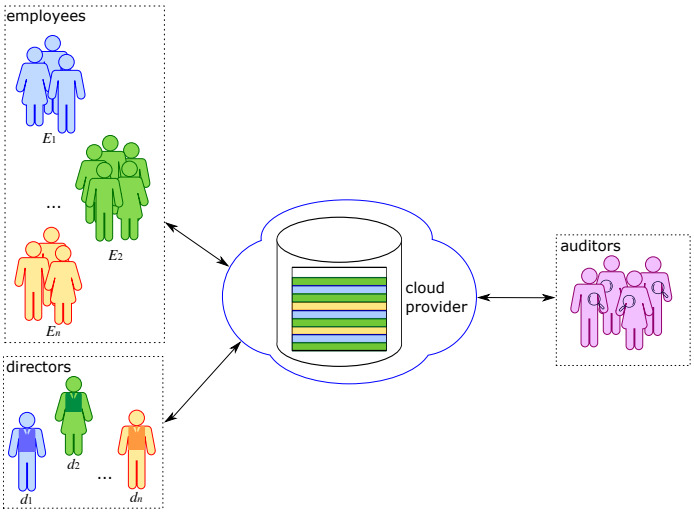

Figure 1. ICA process at an external cloud provider

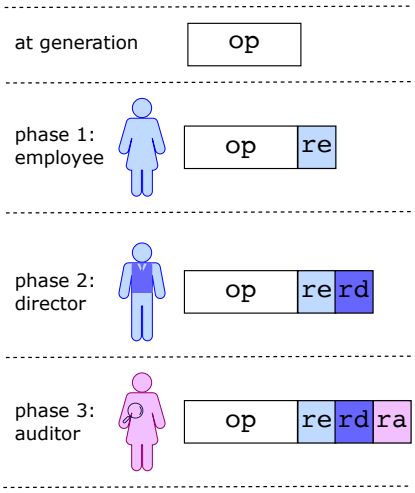

Figure 2. Evolution of the operation record during the ICA process

the ICA process, leveraging selective encryption to provide a self-enforcement layer guaranteeing both confidentiality against the cloud provider and selective visibility of operations and reports to subjects accessing the service. Third, we provide a solution based on compact and effective tags that, together with selective encryption, enable enforcement of write access restrictions on data, dynamically adjusting access privileges with the evolution of the ICA process. The main advantage of our solution is the direct support of the ICA process in an effective and easy way, leveraging the basic services of the cloud provider.

The remainder of this paper is organized as follows. Section 2 presents the scenario, the main concepts, and the requirements of the ICA process. Section 3 presents a formalization of the ICA process, modeling the different concepts and the enforcement of the requirements in terms of access regulations. Section 4 introduces the organization of storage for outsourcing and the basic elements of our approach. Sections 5 and 6 present the realization of the ICA process via selective encryption and tag management. Section 7 illustrates the pseudocode of the functions implementing our approach. Section 8 proves the correctness of our approach for enforcing the ICA process. Section 9 discusses related work. Finally, Section 10 concludes the paper.

2 Scenario and Problem Statement

We consider an organization composed of *units* whose *employees* process and manage different *operations*. Each unit is under the control of a unit *director*, who is responsible for the activity of the unit. For simplicity of exposition, but without loss of generality, we assume that each director is responsible for one unit only. As an example, which is also the scenario that inspired our work, the organization can be a bank and the units the branches of the bank. Each branch has one director and some employees. Independent auditors - appointed by the bank - oversee the working of all the branches of the bank. Operations are the different transactions processed at each branch of the bank (e.g., cash deposit, credit card payment, cheque deposit).

All operations must undergo an *Internal Controls and Audit* (ICA) process to ensure that they are compliant with internal rules as well as with laws to prevent fraud. For each operation, the ICA process requires the execution of three phases, illustrated in Figure 2. The first phase, by an employee of the unit, typically the same who processes the operation, produces the employee report (denoted `re`) on the operation. The second phase, by the unit's director, produces the director report (denoted `rd`) on the operation and its `re`. Finally, the third phase, by an auditor, controls the whole chain and produces the auditor report (denoted `ra`) on the operation and its `re` and `rd`.

Rules regulating the ICA process can be summarized with the following requirements.

R1 - Operation and report visibility. The information about each operation and its ICA reports should be accessible only to the employees and director of the unit in charge of managing the operation and to the auditors.

R2 - Report generation. A report on an operation can be generated only by subjects with the role (employee, director, or auditor) for the report. A report can be updated only by the individual (employee, director, or auditor) who created it and only until its commitment. The director (auditor, resp.) phase can start only when the employee (director, resp.) phase has completed.

R3 - Report integrity and accountability. Tampering of ICA reports should not go undetected and reports cannot be repudiated.

In the next section, we first formalize the ICA process assuming its execution within the trusted organization's domain, and then illustrate our approach for outsourcing the whole operation management and ICA process to an external cloud provider.

3 ICA Process Modeling

The main concepts to be captured for the ICA process management are the units of the organization, with its employees and directors, the auditors, as well as the operations and the different reports on them. We model units as a set $U = \{u_1, \ldots, u_n\}$, where each unit $u \in U$ is in charge of a set

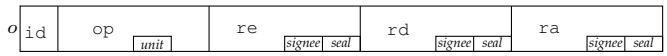

Figure 3. Structure of the record corresponding to an operation

O_u of operations and comprises a set $E_u = \{e_1, \ldots, e_m\}$ of employees and a director d_u, with $d_u \notin E_u$. We denote the unit where employee e (director d, resp.) works with *unit*(e) (*unit*(d), resp.). Auditors are modeled as a set $A = \{a_1, \ldots, a_z\}$. Notations O, D, and E are used to refer to the set of operations, directors, and employees of all units, respectively (i.e., given the set $U = \{u_1, \ldots, u_n\}$ of units, $D = \{d_1, \ldots, d_n\}$, $E = E_{u_1} \cup \ldots \cup E_{u_n}$, and $O = O_{u_1} \cup \ldots \cup O_{u_n}$). Notation S=$E \cup D \cup A$ denotes the set comprising all subjects (i.e., employees, directors, and auditors). With A denoting the set of auditors, with a slight abuse of notation, we will use $\{A\}$ to denote the set comprising symbol A.

Example 3.1. Our running example refers to a bank with two units (branches) U=$\{X, Y\}$, where unit X has employees E_X=$\{x_1, x_2, x_3\}$ and is directed by d_X, and unit Y has employees E_Y=$\{y_1, y_2\}$ and is directed by d_Y. Sets O_X and O_Y denote the operations processed at units X and Y, respectively. The set of independent auditors is A=$\{a_1, a_2\}$.

We model operations in O as records with multiple fields. Figure 3 illustrates the fields of the operations, also highlighting their sub-fields of interest for our modeling. In particular, each operation o comprises the following fields: `id`, reporting the operation identifier; `op`, corresponding to the operation content, including the identifier of the unit where the operation has been processed, denoted *unit*(o); `re`, `rd`, `ra`, corresponding to the employee, director, and auditor report, respectively. The reports, initialized as `null` at the beginning of the process, are filled as the process advances. In particular, each report `r` includes the information on the subject that produced it (*signee*(`r`)) and a signature produced by the subject at report completion (*seal*(`r`)), which have value `null` at the beginning of the process. In the following, we will use the classical dot notation (e.g., o.`op`) to refer to a specific field of an operation record, and $o.*$ to refer to all fields of the operation. Also, for simplicity, we will use the term *operation* to refer interchangeably to the operation and its record. Figure 4 summarizes the notation used in the paper.

The generation of a report is modeled as a write operation on the corresponding report field. Also, when the report is taken in charge by a subject, the corresponding signee sub-field is initialized to the subject's identity. A phase is considered completed for an operation when the corresponding record is signed (i.e., the seal sub-field takes a value.)

The enforcement of the requirements in the previous section can be easily captured by controlling read and write actions on the different fields of an operation and considering as authorized only those actions that satisfy the requirements.

The set AUTH of actions to be authorized according to each requirement is defined as follows.

U	set of units
E	set of employees of the organization
D	set of directors of the organization
A	set of independent auditors
S	set of subjects ($E \cup D \cup A$)
O	set of operations processed at the organization
u	unit
E_u	set of employees of unit u
O_u	set of operations processed at unit u
d_u	director of unit u
o.`id`	operation identifier
o.`op`	operation content
o.`re`	employee report of operation o
o.`rd`	director report of operation o
o.`ra`	auditor report of operation o
o.`te`	tag regulating write of `re`
o.`td`	tag regulating write of `rd`
o.`ta`	tag regulating write of `ra`
o.`tp`	tag regulating the evolution of the ICA phases
unit(o)	unit where operation o has been processed
signee(o.`r`)	creator of report `r`∈{`re`,`rd`,`ra`}
seal(o.`r`)	signature for the integrity of report `r`∈{`re`,`rd`,`ra`}

Figure 4. Notation used in the paper

R1 - Operation and report visibility. $\forall s \in S$, $\forall o \in O_u$, $\forall u \in U$:
`read`(s, $o.*$)∈AUTH iff ($s \in E_u \cup \{d_u\} \cup A$).

R2 - Report generation. $\forall s \in S$, $\forall o \in O_u$, $\forall u \in U$:
`write`(s, o.`re`)∈AUTH iff
$(s \in E_u \wedge \mathit{signee}(o.\texttt{re}) = \texttt{null}) \vee$
$(s = \mathit{signee}(o.\texttt{re}) \wedge \mathit{seal}(o.\texttt{re}) = \texttt{null})$;
`write`(s, o.`rd`)∈AUTH iff
$\mathit{seal}(o.\texttt{re}) \neq \texttt{null} \wedge s = d_u \wedge \mathit{seal}(o.\texttt{rd}) = \texttt{null}$;
`write`(s, o.`ra`)∈AUTH iff
$(\mathit{seal}(o.\texttt{rd}) \neq \texttt{null} \wedge s \in A \wedge \mathit{signee}(o.\texttt{ra}) = \texttt{null}) \vee$
$(s = \mathit{signee}(o.\texttt{ra}) \wedge \mathit{seal}(o.\texttt{ra}) = \texttt{null})$.

It is easy to see the correspondence (and equivalence) of formalization above with the informal requirements stated in the previous section. In particular, **R1** considers authorized to view an operation and its reports all and only the employees and director of the unit to which the operation pertains, and the auditors. **R2** restricts authorized write actions on reports as follows. An employee report that has not started yet (signee is `null`) can be written by any employee of the unit to which the operation pertains, while if started it can be written only by the employee who has started it, provided the employee phase has not completed (i.e., the seal of the employee report is `null`). A director report can be written only if the employee phase has been completed (employee report is sealed), only by the director of the unit to which the operation pertains, and provided the director phase has not completed (i.e., the seal of the director report is `null`). Finally, an auditor report can be written only if the director phase has completed (director report is sealed). If the auditor report has not started yet (signee is `null`), it can be written by any auditor, while if started only by the auditor who has started it, and provided the auditor phase has not completed (i.e., the seal of the auditor report is `null`).

Requirement **R3 - Report integrity and accountability** does not need specific control on the action, and can be simply guaranteed by having subjects signing a report when

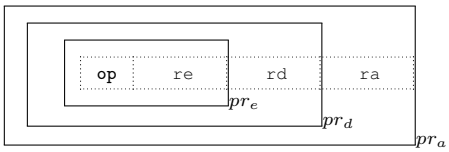

Figure 5. Layered structure of signatures in the ICA process

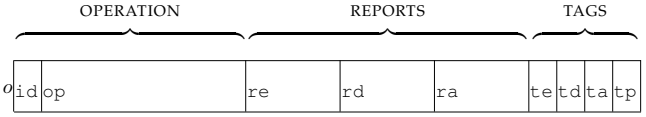

Figure 6. Structure of outsourced operation record

complete (seal field explained above). To this end, each subject s (employee, director, and auditor) has a private-public key pair. The signature of a subject on their report is computed by encrypting (with the subject's private key) the result of a hash function on the concatenation of the signature of the previous phase (or the operation field for the employee's phase) and the report. More precisely, let o be an operation, and e, d, and a be the employee, director, and auditor producing reports for it. Denoting with pr_s the private key of subject s, the three signatures associated with the reports at each step are then computed as: *seal*(re)=*E*(*h*(op||re), pr_e); *seal*(rd)=*E*(*h*(*seal*(re)||rd), pr_d); *seal*(ra)=*E*(*h*(*seal*(rd)||ra), pr_a). Each signature guarantees the integrity of the corresponding report (and of the information on which it was computed) and makes the subject who produced the report accountable for its content. Figure 5 shows the layered organization of the signatures. With this note on requirement **R3**, we consider it as part of the internal processing and hence discard it from further consideration.

4 ICA PROCESS OUTSOURCING

While the formalization, and subsequent enforcement, of access control for the ICA process within the trusted organization setting is relatively simple, it bears several complications when outsourced to an external - not fully trusted - provider. As a matter of fact, the external provider, while trustworthy for correctly enforcing the services requested, should not be allowed to know the content of the operations and of their reports. Also, while offering storage and access functionality, the provider cannot be fully trusted by itself for access control enforcement.

Our approach to delegate storage and management to the external provider while ensuring protection of data confidentiality and control of accesses relies on the use of *selective encryption* [3] dynamically applied to the data and their reports. Such selective encryption offers a self-protection layer to operations and reports, guaranteeing their confidentiality against the provider while enabling enforcement of access control. Selective encryption is based on symmetric encryption and has then a limited overhead. Also, our selective encryption leverages a hierarchical organization of encryption keys, so to enable the release of a single key to each subject involved in the process. The hierarchy of keys reflects the organizational structure of the ICA process, enabling subjects to derive, from their own key, other keys needed for the process. The external provider collaborates in the execution of the ICA process, while remaining agnostic with respect to it.

We start by defining the organization of operations and of reports for external storage and by introducing the hierarchical organization of encryption keys.

Operations and reports. Operations and their reports are stored at the external provider as records of a relation $\mathbb{O}$ encrypted at the field level. In addition to the operation content and reports, each operation record also contains some *tags* used for regulating write actions on the operation reports. In particular, as illustrated in Figure 6, for each operation, in addition to an operation identifier (id), the following fields (all encrypted) are maintained: the operation content (op); its associated reports (re, rd, ra); and four control tags, three for regulating write actions on the corresponding reports (te, td, ta) and one (tp) for enforcing the different phases of the ICA process.

Operation and report visibility (**R1**), as well as report generation (**R2**), will be managed through the (agnostic) provider simply as read/write accesses on the different fields protected through encryption, which provides a self-protection layer and enforces the access rules.

Key hierarchy. Our approach relies on the use of symmetric encryption and on a hierarchical organization of encryption keys, which supports the derivation of keys from other keys and from some public information (*tokens*). Each symmetric key has its secret value k and a public label l. Key derivation is based on the use of precomputed public tokens between keys [4]. Given two keys k_i and k_j, a public token $t_{i,j}$ is computed as $t_{i,j}$=$k_j \oplus h(k_i,l_j)$, where h is a deterministic cryptographic function, $\oplus$ is the bitwise xor operator, and l_j is the publicly available label associated with key k_j. The existence of public token $t_{i,j}$ allows any subject knowing k_i to derive key k_j through token $t_{i,j}$ and public label l_j. Sequential application of tokens enables the derivation of sequences of keys. Tokens, like key labels, need not to remain secret and can be stored at the external provider. In the following, for simplicity, we will use k to denote a key k, its label l, or the pair $\langle k, l\rangle$ when clear from the context. Also, we denote with $k_i \rightsquigarrow k_j$ the existence of a sequence of tokens enabling the derivation of k_j from k_i.

The enforcement of the requirements introduced in Section 2 and formalized in Section 3 is then realized through the application of different encryption keys to the different fields of the operation records. We focus first on the operations and reports visibility (**R1**) to be guaranteed to the different subjects of the ICA process. We then address report generation (**R2**) and management of the process.

5 OPERATIONS AND REPORTS VISIBILITY

The encryption of operations and their reports provides a self-enforcement layer of protection guaranteeing operation and report confidentiality against the external provider. Also, selective application of encryption, meaning the use of different encryption keys for different operations and their different fields, enables the enforcement of access control

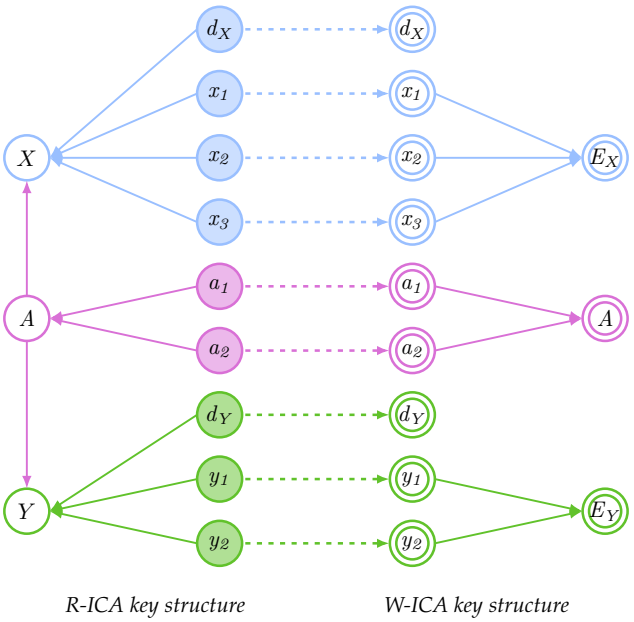

Figure 7. Read and write ICA key structure for Example 3.1

restricting plaintext visibility of operations and reports only to those subjects who know the corresponding key.

We leverage the hierarchical key organization to enable each subject (employee, director, and auditor) involved in the ICA process to derive keys as needed starting from their own individual key. To this end, we define the key encryption hierarchy based on the organization of the ICA process. Our ICA-based key derivation structure for supporting read actions, which we call *R-ICA key structure*, is defined as follows.

Definition 5.1 (R-ICA key structure). Let U, E, D, and A be the set of units, employees, directors, and auditors respectively. The *R-ICA key structure* for the ICA process is defined as a triple $\langle K,T,\phi\rangle$, with K a set of keys and T a set of tokens such that there is:

- a key $k_s \in K$ for each subject $s \in E \cup D \cup A$;
- a key $k_u \in K$ for each unit $u \in U$;
- a key $k_A \in K$ for the set A of auditors;
- a token $\langle k_s, k_u\rangle$ from each k_s to k_u, with $s \in E \cup D$, and u=*unit*(s);
- a token $\langle k_a, k_A\rangle$ from each k_a to k_A, with $a \in A$;
- a token $\langle k_A, k_u\rangle$ from k_A to k_u for each $u \in U$;

and function $\phi : U \cup E \cup D \cup A \cup \{A\} \rightarrow K$ such that $\phi(x)$=k_x for each x in the domain of the function.

The R-ICA key structure can be represented as a directed acyclic graph where nodes correspond to keys and arcs to tokens. Figure 7 graphically illustrates, on the left hand-side of the graph, the R-ICA key structure of our running example. For simplicity, in the figure, we denote each key k_i simply with the entity i to which it refers (e.g., X denotes key k_X of unit X).

Intuitively, the R-ICA key structure enables each employee $e \in E_u$ and director d_u to derive the key of their unit u, denoted $k_e \rightsquigarrow k_u$ and $k_{d_u} \rightsquigarrow k_u$, respectively (e.g., $k_{d_X} \rightsquigarrow k_X$). Each auditor can directly derive key k_A shared among the set of auditors, and can indirectly derive the key of each of the units (e.g., $k_{a_i} \rightsquigarrow k_A \rightsquigarrow k_X$). This ability of subjects to derive keys of units is clearly visible from the paths connecting nodes representing the keys of the subjects to nodes representing the keys of the units in Figure 7. In the figure, nodes (leaves of the R-ICA key structure) with colored background denote the keys released to subjects, while all other keys are derived.

The selective application of encryption to operations and reports, besides protecting their confidentiality against the provider, enables to control visibility against other subjects. As a matter of fact, the encryption of a field with a key k makes the field accessible (and hence its content visible) only to subjects able to derive k. Hence, enforcement of the visibility requirement (**R1**) corresponds to dictating the encryption of operations and reports to be with the key associated with the unit to which the operation refers. By construction, such key is derivable only by the unit director and employees, and all the auditors.

Denoting with function $\lambda(o.f)$ the association between a field f of an operation record o and the key with which it is encrypted, **R1** translates then to requesting satisfaction of the following property.

Property 5.1 (Encryption key assignment). Let $\langle K,T,\phi\rangle$ be a R-ICA key structure. Function $\lambda(o.f)$ assigning the key to be used for encrypting each field $f \in$ {`op`,`re`,`rd`,`ra`} of each operation $o \in O$ is defined as $\lambda(o.\texttt{op})=\lambda(o.\texttt{re})=\lambda(o.\texttt{rd})=\lambda(o.\texttt{ra})=\phi(unit(o))$.

In other words, the operation and all its reports are encrypted with the key of the unit where the operation has been processed.

The definition of the R-ICA key structure for key derivation (Definition 5.1) and the adoption of function λ for identifying the key to be used for encrypting an operation and its reports (Property 5.1) guarantee the satisfaction of requirement **R1**. Indeed, all operations processed in a unit and their reports are encrypted with the unit key that the authorized subjects (i.e., the employees and director of the unit as well as all the auditors) can derive through the R-ICA key structure (see Section 8, Theorem 8.1).

6 Report Generation

The encryption of operations and reports illustrated in the previous section naturally and simply enforces the visibility requirement (**R1**) making operations and reports visible to all and only authorized subjects.

The regulation of reports generation (**R2**), and hence of the write actions on the report fields of operations, cannot be enforced only with the encryption described in the previous section. In fact, not all the readers of a report are authorized to write it. Also, write authorizations change during the ICA process. Hence, the selective encryption of reports enforced for regulating visibility falls short for enforcing dynamically changing write authorizations. Our approach to enforce requirement **R2** leverages the use of tags associated with reports, where each tag represents an (again selectively) encrypted randomly generated secret. To be allowed to write a report, subjects need to prove to be able to decrypt the associated tag. The control to check the subject ability to decrypt tags relies on the cooperation of the provider,

which remains however agnostic of the process as well as of the operation and report contents. Clearly, since the verification of tags for regulating write authorizations is with the provider, and such a verification requires knowledge of the key with which tags are encrypted, keys used for encrypting tags cannot be the same ones used for regulating visibility of operations and reports.

We first introduce the keys to be used for encrypting tags, and then elaborate more on the control performed by the provider on tags and on the management of tags.

6.1 W-ICA key structure

Keys for encrypting tags are designed similarly to the ones for encrypting operations and reports, introducing a hierarchical key structure formally defined as follows.

Definition 6.1 (W-ICA key structure). Let $U, E, D,$ and A be the sets of units, employees, directors, and auditors respectively, and $\mathbb{P}$ be the provider. The *W-ICA key structure* is defined as a triple $\langle K',T',\phi'\rangle$, with K' a set of keys and T' a set of tokens such that there is:

- a key k'_s for each subject $s \in E \cup D \cup A \cup \{\mathbb{P}\}$;
- a key k'_{E_u} for the set E_u of employees of each unit $u \in U$;
- a key k'_A for the set A of auditors;
- a token $\langle k'_s, k'_{E_u}\rangle$ from each k'_s to k'_{E_u}, with $s \in E_u$;
- a token $\langle k'_a, k'_A\rangle$ from each k'_a to k'_A, with $a \in A$;
- a token $\langle k'_{\mathbb{P}}, k'_s\rangle$ from $k'_{\mathbb{P}}$ to k'_s for each $s \in E \cup D \cup A$;

and function $\phi' : \{E_u : u \in U\} \cup E \cup D \cup A \cup \{A\} \cup \{\mathbb{P}\} \rightarrow K'$ such that $\phi'(x)=k'_x$ for each x in the domain of the function.

It is easy to see the correspondence between the W-ICA key structure and the R-ICA key structure illustrated in the previous section. Apart for the presence of the provider as a subject, the only minor difference between them is that units keys (derivable in the R-ICA key structure by employees, directors, and auditors), are replaced by keys derivable by employees only, to support write actions restricted to them (e.g., E_X instead of X and E_Y instead of Y).

Figure 7 graphically illustrates, on the right hand-side of the graph, the W-ICA key structure of our running example.[1] In the figure, nodes of the W-ICA key structure are denoted with a double circle, to distinguish them from the nodes of the R-ICA key structure. For simplifying key management, and maintaining the release to each subject of one key only, tokens are also defined enabling the derivation of k'_s from the corresponding k_s for each subject $s \in E \cup D \cup A$ (dotted lines connecting nodes in the R-ICA key structure to the nodes in the W-ICA key structure in Figure 7). Hence, each subject s will be able to derive all the keys needed for operating from their own key (i.e., k_s in the R-ICA key structure, corresponding to the nodes with colored background in Figure 7).

1. For readability of the figure, the node corresponding to the key of the provider, which is connected with a token to each subject's key of the W-ICA key structure is omitted.

6.2 Write control and tags encryption

As anticipated, our approach to enforce requirement **R2** leverages the use of tags. Each operation has four tags, one for each report (i.e., `te`, `td`, and `ta` for `re`, `rd`, and `ra`, resp.) and one for keeping track of the phase evolution of the ICA process (i.e., `tp`). Tag encryption enables regulating write actions on the reports. Intuitively, a subject will be allowed to write a report (or the tag associated with it) only if the subject proves to be able to correctly decrypt both the tag associated with the report (which enforces the control on the subject having the proper role for it) as well as the phase tag (which enforces the control of the write action occurring in the right phase of the ICA process).

Let $\lambda'(o.\texttt{tag})$ be the key that permits to decrypt a `tag` of operation o (on which we elaborate in the next sub-section), and W be the set of write actions accepted by the provider. The regulation of write actions on reports via the use of encrypted tags can be enforced by having the provider accepting only write actions that pass tag verification, as captured by the following definition.

Definition 6.2 (Write control (reports)). Let $\langle K',T',\phi'\rangle$ be a W-ICA key structure, and λ' be a function assigning keys to tags. A write action by subject $s \in S$ on a report field $f \in \{\texttt{re}, \texttt{rd}, \texttt{ra}\}$ of an operation $o \in O$ is accepted iff the following conditions, depending on the specific field, hold:

- `write`(s,o.`re`) $\in$ W
  iff $\phi'(s) \rightsquigarrow \lambda'(o.\texttt{te}) \wedge \phi'(s) \rightsquigarrow \lambda'(o.\texttt{tp})$;
- `write`(s,o.`rd`) $\in$ W
  iff $\phi'(s) \rightsquigarrow \lambda'(o.\texttt{td}) \wedge \phi'(s) \rightsquigarrow \lambda'(o.\texttt{tp})$;
- `write`(s,o.`ra`) $\in$ W
  iff $\phi'(s) \rightsquigarrow \lambda'(o.\texttt{ta}) \wedge \phi'(s) \rightsquigarrow \lambda'(o.\texttt{tp})$.

In other words, a subject will be allowed to write a report if and only if the subject proves to be able to decrypt the tag of the report and the phase tag (i.e., the subject can derive the keys for their decryption, denoted $\phi'(s) \rightsquigarrow \lambda'(tag)$).

While simply stated in the definition above, the control of write actions via tag encryption is not trivial in its enforcement. The complication arises from the need to support different write authorizations on the different reports of the same operation, and from the dynamic nature of such authorizations that evolve through the ICA process (Section 3). This implies that the key used to encrypt tags (i.e., function λ') needs to change during the ICA process. For instance, an employee report can be written by any employee before the first phase starts, but when an employee (internally recorded in the report as signee) takes it in charge, only this employee can be authorized to modify it. Similar considerations hold for the auditor report. Also, the encryption of the phase tag needs to evolve, to enable execution of write actions only within the correct phase and preventing subjects to operate on a report outside (before or after) the corresponding phase. In fact, no report (director or auditor) can be started if the previous phase has not been completed, that is, its report (employee or director) has not been sealed, and no report can be modified after its corresponding phase is concluded (internally recorded in the report as sealed). While the enforcement of the control within the organization (Section 3) can make reference to the signee and seal recorded in each report, such information is

clearly not visible to the agnostic provider, which cannot know the content of the report and does not even need to know, nor should have understanding of, the ICA process. The evolution of phases should then be reflected by a corresponding evolution of the phase tag encryption.

In summary, the provider can only enforce basic (agnostic) controls on write actions on report fields as per Definition 6.2. Hence, not only the tags for a same operation need to be encrypted with different keys, but also such encryption - or more precisely the key (λ') associated with them - as well as the one of the phase tag need to change with the evolution of the phases of the ICA process.

More formally, the enforcement of requirement **R2** through tag encryption and the controls of Definition 6.2, requires tags encryption to follow the phases of the ICA process, as formalized by the following property.

Property 6.1 (Tag encryption). Let $\langle K',T',\phi' \rangle$ be a W-ICA key structure. Function $\lambda'(o.\texttt{tag})$ assigning the key to be used for encrypting each tag $\texttt{tag} \in \{\texttt{te},\texttt{td},\texttt{ta},\texttt{tp}\}$ of each operation $o \in O$ is such that:

1) $signee(o.\texttt{re}) = \texttt{null} \iff$
   $\lambda'(o.\texttt{te}) = \lambda'(o.\texttt{tp}) = \phi'(E_{unit(o)})$;
2) $signee(o.\texttt{re}) \neq \texttt{null} \wedge seal(o.\texttt{re}) = \texttt{null} \iff$
   $\lambda'(o.\texttt{te}) = \phi'(signee(o.\texttt{re})) \wedge \phi'(signee(o.\texttt{re})) \rightsquigarrow \lambda'(o.\texttt{tp})$;
3) $seal(o.\texttt{re}) \neq \texttt{null} \wedge seal(o.\texttt{rd}) = \texttt{null} \iff$
   $\lambda'(o.\texttt{td}) = \lambda'(o.\texttt{tp}) = \phi'(d_{unit(o)})$;
4) $seal(o.\texttt{rd}) \neq \texttt{null} \wedge signee(o.\texttt{ra}) = \texttt{null} \iff$
   $\lambda'(o.\texttt{ta}) = \lambda'(o.\texttt{tp}) = \phi'(A)$;
5) $signee(o.\texttt{ra}) \neq \texttt{null} \wedge seal(o.\texttt{ra}) = \texttt{null} \iff$
   $\lambda'(o.\texttt{ta}) = \phi'(signee(o.\texttt{ra})) \wedge \phi'(signee(o.\texttt{ra})) \rightsquigarrow \lambda'(o.\texttt{tp})$.

In other words, following the enumerated list above:

1) at the start of the process for an operation within a unit, the tag of the employee report and the phase tag should be encrypted with the key associated with the set of unit's employees (*initial state, phase 1 can start*);
2) when the report is taken in charge by an employee, its tag should be encrypted with the key of that specific employee, who should also be able to derive the key used for the phase tag (*phase 1 started*);
3) when the employee report is completed, the tag of the director report and the phase tag should be encrypted with the key of the unit's director (*phase 1 completed, phase 2 can start/started*);
4) when the director report is completed, the tag of the auditor report and the phase tag should be encrypted with the key of the auditors (*phase 2 completed, phase 3 can start*).
5) when the auditor report is taken in charge by an auditor, its tag should be encrypted with the key of that specific auditor, who should also be able to derive the key used for the phase tag (*phase 3 started*).

Property 6.1 dictates the conditions that should hold to enforce the write control in Definition 6.2 correspond to the authorization control on write actions as it would be internally enforced as discussed in Section 8 (see Theorem 8.2).

We next illustrate the management of tags, namely their generation, encryption, and evolution to ensure satisfaction of Property 6.1.

6.3 Tag management

As said, tags are randomly generated secrets which are encrypted with a key whose knowledge regulates write actions. Basically, according to Definition 6.2, for a subject to be able to write a report, the subject needs to be able to decrypt both the tag of the report and the phase tag, meaning the tags must be encrypted with keys derivable by the subject.

Let us first concentrate on report tags, we will then address phase tags. For simplicity, we refer the discussion to a single operation at a given unit, following its evolution through the ICA process. The enumerated items in the discussion refer to the items in Property 6.1

Report tags. As per Property 6.1, to enable the start of each phase (i.e., the creation of the respective report), report tags should be encrypted as follows. For enabling start of phase 1 (initial state), `te` should be encrypted with the key of the set of the unit's employees (item 1); for enabling start of phase 2, `td` should be encrypted with the key of the unit's director (item 3); and for enabling start of phase 3, `ta` should be encrypted with the key of the set of auditors (item 4). Although the control on such tags is enforced (and has effect) when the respective phase starts, tags need not to be generated on the fly, but can be generated in advance. Generating tags in advance has the great advantage of not requiring any synchronization between subjects. In particular, the employee who processes an operation does not need to wait for tags generation by the other subjects, who do not need to be online and available when the operation is processed. We also note that, while the random values for the different tags (i.e., `te`, `td`, and `ta`) need to be different (else the protection by encryption could be bypassed), all the operations of the same unit could even have the same triple of tag values (since at start all operations of a same unit are subject to the same authorizations). We then simply assume that, for each unit, three tag values, one for each tag (`te`, `td`, `ta`) are precomputed (randomly generated and encrypted). Such precomputed tags will be attached to all operations of the unit. Note that such precomputation of tags does not require that auditors be defined a priori, but only the existence of a key (that will be) known to and associated with them and not known to employees and directors.

Tags associated with employee (auditor, resp.) reports need also to evolve during the process, and their encryption change when a report is taken in charge by a specific employee (auditor, resp.). In particular, `te` (`ta`, resp.) needs to become restricted only to the specific employee (auditor, resp.) who started `re` (`ra`, resp.), as captured in item 2 (item 5, resp.) of Property 6.1. This evolution requires the employee (auditor, resp.) to overwrite the corresponding tag which should become encrypted with the employee (auditor, resp.) individual key. Note that, not only the encryption key associated with the tag needs to change but also the encrypted randomly generated value needs to be regenerated (as otherwise the re-encryption could clearly be easily bypassed, since all employees/auditors would potentially know the value behind the encryption). Note that this rewriting is not requested in the director phase, since the director is only one for each unit and hence authorizations do not need to be restricted when the report is started.

		te	td	ta	tp
at generation		E_u	d_u	A	$E_u(d_u(A))$
phase 1: employee	e starts re	e •	d_u	A	$E_u(d_u(A))$
	e seals re	e	d_u	A	$d_u(A)$
phase 2: director	d_u starts rd	e	d_u	A	$d_u(A)$
	d_u seals rd	e	d_u	A	A
phase 3: auditor	a starts ra	e	d_u	a •	A
	a seals ra	e	d_u	a	*dummy*

Figure 8. Evolution of keys associated with tags for operation (• denotes that the random value is also refreshed)

Phase tag. The phase tag needs a separate discussion. In principle, its initial setting for enabling start of phase 1 could be simple: it should be encrypted with the key of the set of unit's employees (item 1). However, as a complicating factor, its evolution cannot be carried out by the subject currently operating like done for the report tags, since the key with which the tag should evolve is the one of the subject(s) in charge of the next phase. More precisely, when an employee completes the employee phase, the phase tag should become encrypted with the key of the director (item 3). Similarly, when a director completes the director phase, the phase tag should become encrypted with the key of the set of auditors (item 5). To accommodate such evolution without requiring any runtime action or synchronization between subjects, phase tags are created with a layered encryption process, where the different layers correspond to the (keys of the) phases in which the tag should evolve. Phase tags are therefore randomly generated secrets encrypted with the key of the auditors, then with the key of the director, and finally with the key of the set of employees. Again, note that this application of different layers of encryption with different keys does not require the different subjects to be available at the same time nor the set of auditors to be defined a priori. It is sufficient to start from a randomly generated value encrypted with a key (that will be) known only to the auditors, encrypt it with the key of the director, and then with the key of the set of employees. This process can be performed in advance again generating a pool of tags from which a phase tag to be associated with an operation can be extracted (and deleted from further consideration) when an operation is created. Evolution of the phase tag is then simply enforced by having a subject completing a phase peeling off the outermost encryption layer, hence making the phase tag encrypted with the key of the subject(s) authorized to perform the next phase. Note that random values used to generate the phase tags need to be all different for the different operations. This is needed to prevent the possibility that the phase tag of an operation could be used for acting on another operation outside the proper phase. Since a phase tag can be decrypted, during each phase, only by subjects authorized for writing the report of the phase, one could think that the phase tag can make the other tags unnecessary. This is not the case since, unlike report tags, phase tags cannot be regenerated by considering new random values to restrict access to the individual employee (auditor, resp.) who takes in charge the report at the start of their respective phases. Due to their layered structured, phase tags can be only peeled off.

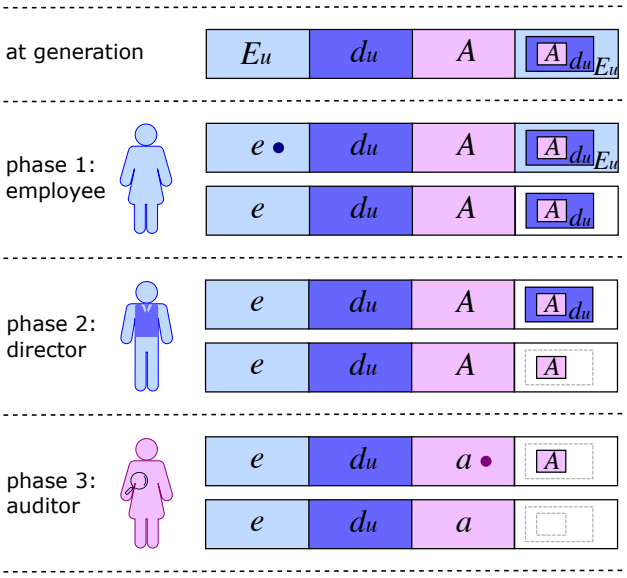

Figure 9. Evolution of the tags during the ICA process (• denotes that the random value is also refreshed)

Tag evolution. Figures 8 and 9 illustrate the evolution of tags by reporting the keys associated with them as the phases of the ICA process evolve. A bullet close to a key within a tag field denotes that the random value has been regenerated before encrypting the field with the new key. The layered structure of the phase tag is represented by the application of the different encryptions. As visible from the figures, when a new operation is created, tags associated with it (extracted from the precomputed pools) are such that: te is encrypted with the key k'_{E_u} of the set of employees, td with the key k'_{d_u} of the director, ta with the key of the set of auditors k'_A, and tp has three layers of encryption with the three keys (in reverse order, that is, with k'_{E_u} as external layer). The only write action allowed is then the writing of the employee report and corresponding tag by any of the employees in E_u. As the first phase starts, the employee e who takes in charge the employee report overwrites te with a new randomly generated value encrypted with its own key k'_e. Employee e will then be the only one able to operate on the report. In fact, other employees will not be able anymore to prove knowledge of the key for tag te. Upon completion (and hence signature) of the report, the employee simply overwrites the phase tag tp by peeling off the outermost layer with k'_{E_u}. This exposes now the phase tag as encrypted with director key k'_{d_u}, hence making write actions on the director report (and its tag) now possible. Again, upon completion of its report, the director simply overwrites the phase tag tp by peeling off the new outermost layer with k'_{d_u}. This exposes now the phase tag as encrypted with key k'_A of the set of auditors, hence making write actions on the auditor report (and its tag) now possible for any of the auditors. The auditor a who takes in charge the report overwrites ta with a new randomly generated value encrypted with its own key k'_a. Auditor a will then be the only one able to operate on the report (other auditors will not be able anymore to prove knowledge of the key for tag ta). Upon completion of the report, the auditor simply

overwrites the phase tag nullifying it, which implies no write action will be accepted anymore.

Write actions on tags. The support of tag evolution as explained above implies the need to support write actions also on tags. Again, like for reports, the writings of tags by the subjects will simply translate in write actions on the corresponding field, which will be executed by the (agnostic) provider. Write actions on the tag fields are regulated similarly to write actions on reports, as captured by the following definition.

Definition 6.3 (Write control (tags)). Let $\langle K',T',\phi'\rangle$ be a W-ICA key structure, and λ' be a function assigning keys to tags. A write action by subject $s \in S$ on a tag field $f \in \{\texttt{te},\texttt{td},\texttt{ta},\texttt{tp}\}$ of an operation $o \in O$ is accepted iff the following conditions, depending on the specific field, hold.

- write(s,o.te) $\in$ W iff $\phi'(s) \rightsquigarrow \lambda'(o.\texttt{te}) \wedge \phi'(s) \rightsquigarrow \lambda'(o.\texttt{tp})$;
- write(s,o.td) $\in$ W iff $\phi'(s) \rightsquigarrow \lambda'(o.\texttt{td}) \wedge \phi'(s) \rightsquigarrow \lambda'(o.\texttt{tp})$;
- write(s,o.ta) $\in$ W iff $\phi'(s) \rightsquigarrow \lambda'(o.\texttt{ta}) \wedge \phi'(s) \rightsquigarrow \lambda'(o.\texttt{tp})$;
- write(s,o.tp) $\in$ W iff $\phi'(s) \rightsquigarrow \lambda'(o.\texttt{tp}) \wedge (\phi'(s) \rightsquigarrow \lambda'(o.\texttt{te}) \vee \phi'(s) \rightsquigarrow \lambda'(o.\texttt{td}) \vee \phi'(s) \rightsquigarrow \lambda'(o.\texttt{ta}))$.

In other words, a subject will be allowed to write a report tag only if the subject proves to be able to decrypt it as well as the phase tag. A subject will be allowed to write the phase tag if the subject proves to be able to decrypt the phase tag and one of the report tags (to restrict the writing of the tag to the subject who has completed a phase for the operation).

7 ICA PROCESS EXECUTION

In this section, we provide the pseudocode (Figure 10) executed by the different subjects and the provider to enforce the process illustrated in the previous section. The pseudocode reports the ICA process on operations stored at the provider, with fields initialized at operation generation as discussed above. In particular, for each operation, the values of the report tags (i.e., te, td, and ta) are set to the triple of values computed for the operation's unit, while the value of its phase tag (i.e., tp) is extracted from a pool of pre-computed tags (see Section 6). Figure 10 includes three rows, one row for each phase of the ICA process. For each phase, we report the functions executed by the subject in charge of the phase (i.e., employee, director, auditor) and the functions executed by the provider. For each phase and each of the involved subjects and the provider, there are the functions for starting the phase (**Start*** and **Write_T***, with * in E, A), updating the corresponding report (**Write_Report*** and **Write_R***, with * in E, D, A), and ending the control phase (**End*** and **Complete_*Phase**, with * in E, D, A).[2] In the following, we describe the working of these functions. We will prove their correctness in Section 8.

2. We note that the three functions managing the start of a phase, the write of the corresponding report, and the end of the phase can be (partially) combined. In this case, it is not necessary for the subject to download the operation of interest more than once.

Start phase. The start phase is needed to allow a specific subject to become in charge of the generation (and then on the possible updates) of a report associated with an operation. As a matter of fact, an employee (auditor, resp.) report can be created by any of the employees (auditors, resp.), while its update is to be restricted by the specific employee (auditor, resp.) who has created it. For the director phase, only the director can create and update the director report and therefore the director has already the exclusive control on the director report itself.

To start the employee phase on an operation, employee e (through function **StartE**) downloads the operation from the provider. Then, the employee decrypts both the employee tag te and the phase tag tp, obtaining values σ and $\sigma_{\texttt{ph}}$, necessary to prove the provider that the employee is authorized to write the employee report of the operation. The employee then generates a new employee tag *new_tag* for the operation by encrypting a randomly generated value with their own key k'_e, and invokes function **Write_TE** passing to the provider the operation id, the *new_tag*, σ, and $\sigma_{\texttt{ph}}$.

Upon receiving the invocation of **Write_TE**, the provider first retrieves the operation of interest and decrypts both the employee tag te and the phase tag tp. The provider verifies if the obtained values match with σ and $\sigma_{\texttt{ph}}$ and, if this is the case, it overwrites te with the value received from the employee and updates $\lambda'(o.\texttt{te})$ with the label of the employee's key (i.e., k'_e), thus making the employee report not modifiable by other employees (they cannot decrypt the new tag and do not know the new secret).

The start of the auditor phase (**StartA**) works in a similar way, but it operates on ta: the auditor invokes function **StartA** to retrieve the operation, decrypts ta and tp, computes a new value for ta, and invokes **Write_TA**. Upon receiving the invocation of **Write_TA**, the provider verifies tags ta and tp and, if the check passes, overwrites ta with the value received from the auditor and sets $\lambda'(o.\texttt{ta})$ to the label of the auditor's key.

Write report. During each of the three control phases of the ICA process, the subject who started the phase (employee, director, or auditor) can write (i.e., create or modify) the corresponding report (function **Write_Report***, with * in E, D, A). To this aim, the subject downloads from the provider the operation of interest and decrypts both the tag regulating write access to the report (e.g., te for re) and the phase tag tp and sends the resulting values to the provider, together with the updated (encrypted) report content. The provider, upon receiving the updated report content and the result of tag decryption, retrieves the operation of interest and verifies tags. If the control is successful, the provider overwrites the report (function **Write_R***, with * in E, D, A).

End phase. To conclude an ICA phase (function **End***, with * in E, D, A), the subject in charge of the phase downloads from the provider the operation of interest and decrypts both the tag regulating write access to the report (e.g., te for re) and the phase tag tp and sends the resulting values to the provider. The provider retrieves the operation of interest and verifies tags. If the control is successful, the provider overwrites the phase tag with the result of its decryption, updating $\lambda'(o.\texttt{tp})$ accordingly (function **Complete_*Phase**,

PHASE 1 - EMPLOYEE

e

StartE(*id*)
1: Let o in $\mathbb{O}$ s.t. o.id=*id*
2: σ:=Dec(o.te, k'_{E_u})
3: σ_{ph}:=Dec(o.tp, k'_{E_u})
4: Generate σ' at random
5: *new_tag*:=Enc(σ', k'_e)
6: **Write_TE**(e, *id*, σ, σ_{ph}, *new_tag*)

Write_ReportE(*id*)
1: Let o in $\mathbb{O}$ s.t. o.id=*id*
2: σ:=Dec(o.te, k'_e)
3: σ_{ph}:=Dec(o.tp, k'_{E_u})
4: re:=Dec(o.re, k_u) /* u=*unit*(o) */
5: Update re
6: **Write_RE**(*id*, σ, σ_{ph}, Enc(re, k_u))

EndE(*id*)
1: Let o in $\mathbb{O}$ s.t. o.id=*id*
2: σ:=Dec(o.te, k'_e)
3: σ_{ph}:=Dec(o.tp, k'_{E_u})
4: **Complete_EPhase**(*id*, *unit*(e), σ, σ_{ph})

$\mathbb{P}$

Write_TE(e, *id*, σ, σ_{ph}, *new_tag*)
1: Let o in $\mathbb{O}$ s.t. o.id=*id*
2: **if** σ=Dec(o.te, λ'(o.te)) AND σ_{ph}=Dec(o.tp, λ'(o.tp)) **then**
3: o.te:=*new_tag*
4: λ'(o.te):=k'_e

Write_RE(*id*, σ, σ_{ph}, re)
1: Let o in $\mathbb{O}$ s.t. o.id=*id*
2: **if** σ=Dec(o.te, λ'(o.te)) AND σ_{ph}=Dec(o.tp, λ'(o.tp)) **then**
3: o.re:=re

Complete_EPhase(*id*, u, σ, σ_{ph})
1: Let o in $\mathbb{O}$ s.t. o.id=*id*
2: **if** σ=Dec(o.te, λ'(o.te)) AND σ_{ph}=Dec(o.tp, λ'(o.tp)) **then**
3: o.tp:=σ_{ph}
4: λ'(o.tp):=k'_{d_u}

PHASE 2 - DIRECTOR

d_u

Write_ReportD(*id*)
1: Let o in $\mathbb{O}$ s.t. o.id=*id*
2: σ:=Dec(o.td, k'_{d_u})
3: σ_{ph}:=Dec(o.tp, k'_{d_u})
4: rd:=Dec(o.rd, k_u) /* u=*unit*(o) */
5: Update rd
6: **Write_RD**(*id*, σ, σ_{ph}, Enc(rd, k_u))

EndD(*id*)
1: Let o in $\mathbb{O}$ s.t. o.id=*id*
2: σ:=Dec(o.td, k'_{d_u})
3: σ_{ph}:=Dec(o.tp, k'_{d_u})
4: **Complete_DPhase**(*id*, σ, σ_{ph})

$\mathbb{P}$

Write_RD(*id*, σ, σ_{ph}, rd)
1: Let o in $\mathbb{O}$ s.t. o.id=*id*
2: **if** σ=Dec(o.td, λ'(o.td)) AND σ_{ph}=Dec(o.tp, λ'(o.tp)) **then**
3: o.rd:=rd

Complete_DPhase(*id*, σ, σ_{ph})
1: Let o in $\mathbb{O}$ s.t. o.id=*id*
2: **if** σ=Dec(o.td, λ'(o.td)) AND σ_{ph}=Dec(o.tp, λ'(o.tp)) **then**
3: o.tp:=σ_{ph}
4: λ'(o.tp):=k'_A

PHASE 3 - AUDITOR

a

StartA(*id*)
1: Let o in $\mathbb{O}$ s.t. o.id=*id*
2: σ:=Dec(o.ta, k'_A)
3: σ_{ph}:=Dec(o.tp, k'_A)
4: Generate σ' at random
5: *new_tag*:=Enc(σ', k'_a)
6: **Write_TA**(a, *id*, σ, σ_{ph}, *new_tag*)

Write_ReportA(*id*)
1: Let o in $\mathbb{O}$ s.t. o.id=*id*
2: σ:=Dec(o.ta, k'_a)
3: σ_{ph}:=Dec(o.tp, k'_A)
4: ra:=Dec(o.ra, k_u) /* u=*unit*(o) */
5: Update ra
6: **Write_RA**(*id*, σ, σ_{ph}, Enc(ra, k_u))

EndA(*id*)
1: Let o in $\mathbb{O}$ s.t. o.id=*id*
2: σ:=Dec(o.ta, k'_a)
3: σ_{ph}:=Dec(o.tp, k'_A)
4: **Complete_APhase**(*id*, σ, σ_{ph})

$\mathbb{P}$

Write_TA(a, *id*, σ, σ_{ph}, *new_tag*)
1: Let o in $\mathbb{O}$ s.t. o.id=*id*
2: **if** σ=Dec(o.ta, λ'(o.ta)) AND σ_{ph}=Dec(o.tp, λ'(o.tp)) **then**
3: o.ta:=*new_tag*
4: λ'(o.ta):=k'_a

Write_RA(*id*, σ, σ_{ph}, ra)
1: Let o in $\mathbb{O}$ s.t. o.id=*id*
2: **if** σ=Dec(o.ta, λ'(o.ta)) AND σ_{ph}=Dec(o.tp, λ'(o.tp)) **then**
3: o.ra:=ra

Complete_APhase(*id*, σ, σ_{ph})
1: Let o in $\mathbb{O}$ s.t. o.id=*id*
2: **if** σ=Dec(o.ta, λ'(o.ta)) AND σ_{ph}=Dec(o.tp, λ'(o.tp)) **then**
3: o.tp:=dummy
4: λ'(o.tp):=null

Figure 10. Pseudocode of the ICA process

with * in E, D, A). At the end of the auditor phase, the phase tag is set to *dummy* and λ'(o.tp) is set to null, since the tag cannot be further decrypted. This terminates the ICA process of the operation.

For simplicity, Figure 10 does not explicitly consider the case in which the requesting subject is not authorized for the write action (i.e., the decrypted tags provided do not match with the ones computed by the provider). In this case, the provider denies the write request.

We close this section with a comment on the integration of the mechanisms above with cloud services at the practical level. The integration with cloud services depends on the specific architecture of the cloud solution and many factors come into play. Our approach essentially integrates two layers of protection: the first layer is the selective encryption, the second layer is the tag-based control on write operations. The realization of the first layer is compatible with a software architecture where the client-side component applies encryption/decryption when interacting with the cloud resources, so that the plaintext is only accessible at the client side. This layer is realized client-side, does not see the involvement of the service provider, and guarantees data and report confidentiality against unauthorized users as well as against the provider. The second layer can be realized by enriching the access control module within software architecture at the provider side. This enables its enforcement within the regular access control services and guarantees that write operations are only executed by authorized subjects.

8 Correctness

We formally prove that our approach (illustrated in the previous sections) correctly enforces the requirements of the ICA process discussed in Section 2 and formalized in Section 3. To prove the correct enforcement of requirements **R1** and **R2**, we focus separately on each of them.

To prove that requirement **R1** (Operation and report visibility) is correctly enforced, we demonstrate that the adoption of selective encryption, combined with key derivation (Section 5) enables only authorized subjects to access

operations and reports in plaintext. This is equivalent to prove that only authorized subjects can derive the encryption keys used to encrypt operations and reports, as formally stated by the following theorem.

Theorem 8.1 (Correct enforcement of R1 - Operation and report visibility). Given sets U, E, D, A, and O of units, employees, directors, auditors, and operations, respectively; a R-ICA key structure $\langle K,T,\phi\rangle$ (Definition 5.1); and a function λ (Property 5.1), if $\forall s \in E \cup D \cup A$, $\forall o \in O$, $\forall f \in \{\texttt{op},\texttt{re},\texttt{rd},\texttt{ra}\}$: $\phi(s) \rightsquigarrow \lambda(o.f) \iff \texttt{read}(s, o.f) \in$ AUTH, then requirement **R1** is satisfied.

Proof: Since according to requirement **R1**, $\forall s \in S, \forall o \in O_u$, $\forall u \in U$, $\texttt{read}(s, o.*) \in$ AUTH iff $(s \in E_u \cup \{d_u\} \cup A)$, we need to prove that $\phi(s) \rightsquigarrow \lambda(o.f)$ iff $(s \in E_u \cup \{d_u\} \cup A)$. According to Property 5.1, $\lambda(o.f)=\phi(u)$, $\forall f \in \{\texttt{op}, \texttt{re}, \texttt{rd}, \texttt{ra}\}$. Therefore, $\phi(s) \rightsquigarrow \lambda(o.f)$ is equivalent to $\phi(s) \rightsquigarrow \phi(u)$. According to Definition 5.1, for each unit u, $\phi(u)=k_u$, and for each subject s, $\phi(s)=k_s$. Therefore, $\phi(s) \rightsquigarrow \phi(u)$ is equivalent to $k_s \rightsquigarrow k_u$. According to Definition 5.1, for each $s \in E \cup D$ there is a token $\langle k_s, k_u\rangle$ if *unit*(s)=u, that is, if $s \in E_u \cup \{d_u\}$. Hence, $\phi(s) \rightsquigarrow \lambda(o.f)$. Since for subjects $s \in E \cup D$ there is no other token from k_s in the R-ICA structure, s cannot derive the key of any other unit. According to Definition 5.1, for each $s \in A$, there is a token $\langle k_s, k_A\rangle$, and there is a token $\langle k_A, k_u\rangle$ for each unit $u \in U$. Hence, $\phi(s) \rightsquigarrow \lambda(o.f)$. ∎

To prove that requirement **R2** (Report generation) is correctly enforced, we operate in two steps: *i)* we first prove (Theorem 8.2) that the pseudo-code in Figure 10 satisfies Property 6.1, and *ii)* we then prove (Theorem 8.3) that the evolution of tags according to Property 6.1 guarantees that write actions permitted by the pseudo-code in Figure 10 are all and only those in AUTH, according to the formal definition of requirement **R2**. For the sake of readability, in the proofs of Theorems 8.2 and 8.3, we rely on Figure 11, which illustrates the evolution of tags according to the pseudo-code in Figure 10, and the corresponding evolution of variables signee and seal of reports during the different phases of the ICA process.

Theorem 8.2. Given a set U, E, D, A, and O of units, employees, directors, auditors, and operations, respectively; a W-ICA key structure $\langle K',T',\phi'\rangle$ (Definition 6.1); and a function λ' (Definition 6.2), the functions in Figure 10 satisfy Property 6.1.

Proof: We discuss each phase of the ICA process separately for an operation $o \in O$, with *unit*(o)=u.

Employee phase. The evolution of variable *signee*(o.re) corresponds to the evolution of o.te. The evolution of variable *seal*(o.re) corresponds to the evolution of o.tp.
At initialization time, when *signee*(o.re)=null, tags o.te and o.tp are encrypted with key $\lambda'(o.\texttt{te})=\lambda'(o.\texttt{tp})=k'_{E_u}=\phi'(E_u)$, lines 2-3 in **StartE**. Note that neither $\lambda'(o.\texttt{te})$ nor $\lambda'(o.\texttt{tp})$ are set to k'_{E_u} anywhere in Figure 10. Hence, item 1 in Property 6.1 is satisfied (row "at generation" in the table in Figure 11).
When employee e takes in charge report o.re and *signee*(o.re) becomes e (i.e., *signee*(o.re)≠null and *seal*(o.re)=null), the value of tag o.te is overwritten with a value encrypted with $k'_e=\phi'(e)$, that is, $\lambda'(o.\texttt{te})=\phi'(e)=\phi'(\textit{signee}(o.\texttt{re}))$, lines 5-6 in **StartE**. The value of tag o.tp remains instead unchanged. Since by Definition 6.1 only employees $e \in E_u$ can derive k'_{E_u}, $\phi'(\textit{signee}(o.\texttt{re}))=\phi'(e)=k'_e \rightsquigarrow \lambda'(o.\texttt{tp})=\phi'(E_u)=k'_{E_u}$. Since $\lambda'(o.\texttt{te})$ is updated only in **Write_TE** iff $\lambda'(o.\texttt{te})=\lambda'(o.\texttt{tp})=k'_{E_u}$, $\lambda'(o.\texttt{te})$ cannot be set to $\phi'(\textit{signee}(o.\texttt{re}))$ when *signee*(o.re)≠null. Hence, item 2 in Property 6.1 is satisfied (row "e starts re" in the table in Figure 11).
When employee e completes report o.re and *seal*(o.re) becomes not null, tag o.tp is updated peeling off the external layer. Therefore, $\lambda'(o.\texttt{tp})=\phi'(d_u)=k'_{d_u}$, line 4 in **Complete_EPhase** (row "e seals re" in the table in Figure 11). Note that $\lambda'(o.\texttt{tp})$ can be set to k'_{d_u} only by **Complete_EPhase** when $\lambda'(\texttt{tp})=k'_{E_u}$ and $\lambda'(\texttt{te})=\phi'(e)$. Therefore, $\lambda'(\texttt{tp})$ can be set to k'_{d_u} only if *signee*(o.re)≠null and *seal*(o.re)=null.

Director phase. The evolution of variable *seal*(o.rd) corresponds to the evolution of o.tp.
Tag td is encrypted with key $\lambda'(\texttt{td})=k'_{d_u}=\phi'(d_u)$, line 2 in **Write_ReportD**, and is never modified. When director d_u takes in charge report rd, as noted above, tag tp is encrypted with the same key k'_{d_u}. Therefore, $\lambda'(\texttt{td})=\lambda'(\texttt{tp})=k'_{d_u}=\phi'(d_u)$, satisfying item 3 in Property 6.1 (row "d_u starts rd" in the table in Figure 11).
When director d_u completes report o.rd and *seal*(o.rd) becomes not null, tag o.tp is updated peeling off the external layer. Therefore, $\lambda'(o.\texttt{tp})=\phi'(A)=k'_A$, line 4 in **Complete_DPhase** (row "d_u seals rd" in the table in Figure 11). Note that $\lambda'(o.\texttt{tp})$ can be set to k'_A only by **Complete_DPhase** when $\lambda'(o.\texttt{tp})=k'_{d_u}$ and $\lambda'(o.\texttt{td})=\phi'(d_u)$. Therefore, $\lambda'(o.\texttt{tp})$ can be set to k'_A only if *signee*(o.rd)≠null and *seal*(o.rd)=null.

Auditor phase. The evolution of variable *signee*(o.ra) corresponds to the evolution of o.ta. The evolution of variable *seal*(o.ra) corresponds to the evolution of o.tp.
At initialization time and till *signee*(o.ra)=null, tag o.ta is encrypted with key $\lambda'(o.\texttt{ta})=k'_A=\phi'(A)$, line 2 in **StartA**. As noted above, also tag o.tp is encrypted with key k'_A. Therefore, $\lambda'(o.\texttt{ta})=\lambda'(o.\texttt{tp})=k'_A=\phi'(A)$, satisfying item 4 in Property 6.1 (row "d_u seals rd" in the table in Figure 11).
When auditor a takes in charge report o.ra and *signee*(o.ra) becomes a (i.e., *signee*(o.ra)≠null), the value of tag o.ta is overwritten with a value encrypted with $k'_a=\phi'(a)$, that is, $\lambda'(o.\texttt{ta})=\phi'(a)=\phi'(\textit{signee}(o.\texttt{ra}))$, lines 5-6 in **StartA**. Since $\lambda'(o.\texttt{ta})$ is updated only in **Write_TA** iff $\lambda'(o.\texttt{ta})=\lambda'(o.\texttt{tp})=k'_A$, $\lambda'(o.\texttt{ta})$ cannot be set to $\phi'(\textit{signee}(o.\texttt{ra}))$ when *signee*(o.ra)≠null. The value of tag o.tp remains instead unchanged. By Definition 6.1 only auditors $a \in A$ can derive k'_A, $\phi'(\textit{signee}(o.\texttt{ra}))=\phi'(a)=k'_a \rightsquigarrow \lambda'(o.\texttt{tp})=\phi'(A)=k'_A$. Hence, item 5 in Property 6.1 is satisfied (row "a starts ra" in the table in Figure 11).
When auditor a completes report o.ra and *seal*(o.ra) becomes not null, tag o.tp is set to *dummy*. Therefore, $\lambda'(o.\texttt{tp})=\texttt{null}$, line 5 in **Complete_APhase** (last row 'a seals ra' in the table in Figure 11).

Note that functions **Write_Report*** (with * in E, D, A), which modify the reports content, check the same conditions as functions **StartE*** (with * in E, D, A) and do not modify the keys with which reports and tags are encrypted.

		signee(re)	λ'(te)	*signee*(rd)	λ'(td)	*signee*(ra)	λ'(ta)	*seal*(re)	*seal*(rd)	*seal*(ra)	λ'(tp)
at generation		–	$\phi'(E_u)$	–	$\phi'(d_u)$	–	$\phi'(A)$	–	–	–	$\phi'(E_u)$
phase 1:	e starts re	e	$\phi'(e)$	–	$\phi'(d_u)$	–	$\phi'(A)$	–	–	–	$\phi'(E_u)$
employee	e seals re	e	$\phi'(e)$	–	$\phi'(d_u)$	–	$\phi'(A)$	*sealed*	–	–	$\phi'(d_u)$
phase 2:	d_u starts rd	e	$\phi'(e)$	d_u	$\phi'(d_u)$	–	$\phi'(A)$	*sealed*	–	–	$\phi'(d_u)$
director	d_u seals rd	e	$\phi'(e)$	d_u	$\phi'(d_u)$	–	$\phi'(A)$	*sealed*	*sealed*	–	$\phi'(A)$
phase 3:	a starts ra	e	$\phi'(e)$	d_u	$\phi'(d_u)$	a	$\phi'(a)$	*sealed*	*sealed*	–	$\phi'(A)$
auditor	a seals ra	e	$\phi'(e)$	d_u	$\phi'(d_u)$	a	$\phi'(a)$	*sealed*	*sealed*	*sealed*	–

Figure 11. Correspondence between in-house variables and outsourced tags

Therefore, the considerations discussed above for report generation hold also for report updates (i.e., only authorized subjects can modify reports). ∎

Having demonstrated that the functions in Figure 10 guarantee the satisfaction of Property 6.1, we now prove that the satisfaction of such a property implies the correct enforcement of requirement **R2**, as formalized in the following theorem.

Theorem 8.3 (Correct enforcement of R2 - Report generation). Given a set U, E, D, A, and O of units, employees, directors, auditors, and operations, respectively; a W-ICA key structure $\langle K', T', \phi' \rangle$ (Definition 6.1); and a function λ' (Definition 6.1), if $\forall s \in E \cup D \cup A$, $\forall o \in O$, $\forall r \in \{\texttt{re}, \texttt{rd}, \texttt{ra}\}$: write($s$, $o.r$)∈W $\iff$ write(s, $o.r$)∈AUTH, then requirement **R2** is satisfied.

Proof: We prove that the Theorem holds for each report of an operation $o \in O$, with *unit*(o)=u.

Employee report: According to requirement **R2**, write(s,o.re)∈AUTH iff $(s \in E_u \wedge \textit{signee}(o.\texttt{re}) = \texttt{null}) \vee (s = \textit{signee}(o.\texttt{re}) \wedge \textit{seal}(o.\texttt{re}) = \texttt{null})$. According to Definition 6.2, write(s, o.re)∈W iff $\phi'(s) \rightsquigarrow \lambda'(o.\texttt{te}) \wedge \phi'(s) \rightsquigarrow \lambda'(o.\texttt{tp})$.
Let us first consider the scenario where $(s \in E_u \wedge \textit{signee}(o.\texttt{re}) = \texttt{null})$. By item 1 in Property 6.1, $\textit{signee}(o.\texttt{re}) = \texttt{null}$ implies $\lambda'(o.\texttt{te}) = \lambda'(o.\texttt{tp}) = \phi'(E_u)$. Since $s \in E_u$, s can derive $k'_{E_u} = \phi'(E_u)$, which is used to encrypt both $o.\texttt{te}$ and $o.\texttt{tp}$. Therefore the equivalence holds.
Let us now consider the scenario where $(s = \textit{signee}(o.\texttt{re}) \wedge \textit{seal}(o.\texttt{re}) = \texttt{null})$. By item 2 in Property 6.1, $\textit{signee}(o.\texttt{re}) \neq \texttt{null} \wedge \textit{seal}(o.\texttt{re}) = \texttt{null}$ implies $\lambda'(o.\texttt{te}) = \phi'(\textit{signee}(o.\texttt{re})) \wedge \phi'(\textit{signee}(o.\texttt{re}) \rightsquigarrow \lambda'(o.\texttt{tp})$. Since $s = \textit{signee}(o.\texttt{re})$, s can derive both the keys used to encrypt $o.\texttt{te}$ (key k'_s of subject s) and $o.\texttt{tp}$ (k'_{E_u}). Therefore the equivalence holds.

Director report. According to requirement **R2**, write(s, o.rd)∈AUTH iff $\textit{seal}(o.\texttt{re}) \neq \texttt{null} \wedge s = d_u \wedge \textit{seal}(o.\texttt{rd}) = \texttt{null}$. According to Definition 6.2, write(s, o.rd)∈W iff $\phi'(s) \rightsquigarrow \lambda'(o.\texttt{td}) \wedge \phi'(s) \rightsquigarrow \lambda'(o.\texttt{tp})$.
By item 3 in Property 6.1, $\textit{seal}(o.\texttt{re}) \neq \texttt{null} \wedge \textit{seal}(o.\texttt{rd}) = \texttt{null}$ implies $\lambda'(o.\texttt{td}) = \lambda'(o.\texttt{tp}) = \phi'(d_u)$. Since $s = d_u$, s knows the keys used to encrypt $o.\texttt{td}$ and $o.\texttt{tp}$ (key k'_s of subject s). Therefore the equivalence holds.

Auditor report. According to requirement **R2**, write(s, o.ra)∈AUTH iff $(\textit{seal}(o.\texttt{rd}) \neq \texttt{null} \wedge s \in A \wedge \textit{signee}(o.\texttt{ra}) = \texttt{null}) \vee (s = \textit{signee}(o.\texttt{ra}) \wedge \textit{seal}(o.\texttt{ra}) = \texttt{null})$. According to Definition 6.2, write(s, o.ra)∈W iff $\phi'(s) \rightsquigarrow \lambda'(o.\texttt{ta}) \wedge \phi'(s) \rightsquigarrow \lambda'(o.\texttt{tp})$.
Let us first consider the scenario where $\textit{seal}(o.\texttt{rd}) \neq \texttt{null} \wedge s \in A \wedge \textit{signee}(o.\texttt{ra}) = \texttt{null}$. By item 4 in Property 6.1, $\textit{seal}(o.\texttt{rd}) \neq \texttt{null} \wedge \textit{signee}(o.\texttt{ra}) = \texttt{null}$ implies $\lambda'(o.\texttt{ta}) = \lambda'(o.\texttt{tp}) = \phi'(A)$. Since $s \in A$, s can derive key k'_A used to encrypt both ta and tp. Therefore the equivalence holds.
Let us now consider the scenario where $s = \textit{signee}(o.\texttt{ra}) \wedge \textit{seal}(o.\texttt{ra}) = \texttt{null}$. By item 5 in Property 6.1, $\textit{signee}(o.\texttt{ra}) \neq \texttt{null} \wedge \textit{seal}(o.\texttt{ra}) = \texttt{null}$ implies $\lambda'(o.\texttt{ta}) = \phi'(\textit{signee}(o.\texttt{ra})) \wedge \phi'(\textit{signee}(o.\texttt{ra})) \rightsquigarrow \lambda'(o.\texttt{tp})$. Since $s = \textit{signee}(o.\texttt{ra})$, s can derive both the keys used to encrypt $o.\texttt{ta}$ (key k'_s of subject s) and $o.\texttt{tp}$ (k'_A). Therefore the equivalence holds. ∎

The theorems above prove that our approach correctly enforces requirements R1 and R2. Clearly, the security of the overall architecture relies on the robustness of the cryptographic primitives used and on the assumption that all the subjects involved in the ICA process (i.e., employees, directors, auditors) and the provider are trustworthy and operate according to the functions in Figure 10 during the different phases of the ICA process. For the first aspect, we note that the used cryptographic primitives commonly recognized to be effective in protecting resources. For the second aspect, we note that a possible misbehavior by any of the subjects involved in the process can be detected by any user authorized to access reports through signature verification. For instance, if the provider enables a non-authorized subject to generate or modify a report for an operation, the signature of such a report will reveal the violation of requirement **R2**.

9 Related work

With the widespread adoption of the cloud computing paradigm, data owners can take advantage of the availability of reliable and convenient services at limited economic costs [5], [6], [7]. Relying on external could providers for data storage and management however raises the problem of ensuring proper protection of data and computations over them [2]. Indeed, cloud providers are usually assumed to be *honest-but-curious* (i.e., trusted to manage the data but not with respect to data confidentiality) and hence data confidentiality, integrity, and availability are possibly at risk [8]. In this scenario, data protection usually relies on data encryption at the owner-side before outsourcing (e.g., [9]). Many efforts have then been devoted to the design of solutions for supporting queries and computations

over encrypted data, to delegate expensive elaborations to the cloud provider without revealing sensitive (encrypted) information (e.g., [10], [11], [12]). Since the cloud providers processing data might even be considered not trustworthy, solutions have been also designed for verifying the integrity of data processing results (e.g., [13], [14]).

In this outsourcing scenario, a line of research related to our proposal is the enforcement of access control restrictions. Existing solutions are based on the adoption of Attribute-Based Encryption (ABE) or of selective encryption. ABE is a public key encryption schema that regulates access to resources according to access policies defined over a set of attributes associated with the secret key of users, or vice versa (e.g., [15], [16]). It can be combined with Attribute-Based Signature (ABS) approaches for supporting the management of write privileges (e.g., [17], [18]). While interesting and widely studied, the adoption of public key encryption and of signature schemas makes ABE less efficient than to our proposal, which is based on symmetric encryption. Performance analysis of ABE implementations [19] show the significant computational cost of these algorithms, significantly more expensive than the algorithms used in our proposal. The integration with the architecture of Cloud Services is also less direct for ABE algorithms, which rely on cryptographic primitives unfamiliar to practitioners, as opposed to the well-known primitives adopted in our design. Also, despite recent solutions supporting policy updates (e.g., [20], [21], [22]), the fine-grained management of policy updates remains not easy to manage. Differently from ABE, selective encryption techniques rely on symmetric encryption and translate the authorization policy into an equivalent encryption policy, regulating resource encryption and key distribution to users (e.g., [3], [23], [24]). Selective encryption is possibly combined with key derivation strategies (e.g., [4], [25]), to limit the key management overhead for users. Even if our solution is inspired and presents similarities with these approaches, existing selective encryption solutions cannot be directly adopted when outsourcing the ICA process, due to the peculiarities of the relationship among subjects (reflected in a specific key derivation structure). Also, selective encryption has been extended to regulate write operations through the adoption of digital signatures and/or on HMAC functions (e.g., [26]), which are not suited to a very dynamic scenario characterized by a large number of small resources like the reports of the ICA process.

A problem that presents similarities with the one addressed in this paper is the object-level tracking along supply chains [27]. However, while also supply chains are characterized by data generated during different phases of object-level tracking, moving the management of these data collections to the cloud presents different security issues than the ones characterizing the ICA process, due to the peculiar roles of the actors interacting in the two scenarios [28]. Selective encryption has been adapted to supply chain management, but there is no need to dynamically regulate write access privileges [29].

A preliminary version of this work was first proposed in [1]. In this paper we have considerably revised and extended the work. We have now provided a formalization of the ICA process, with formal definitions for the requirements and the process modeling. The proposed approach for outsourcing data and operations to cloud providers is also now formally defined and accompanied by properties and proofs of correctness. Also, the approach has been improved with a new tag-based approach for controlling write actions, based on a more efficient and slim solution with precomputed tags and the inclusion of a layered phase tag supporting the evolution of the process.

10 Conclusions

We presented an approach enabling organizations to securely rely on cloud-based services for performing corporate governance Internal Controls and Audit functions. By leveraging hierarchical selective encryption built on the organizational structure, our approach provides a self-protection layer to data and their reports for outsourcing, which guarantees both confidentiality against the cloud provider and support for access regulation. Report management is also conveniently enforced with the support of the cloud provider via compact tags similarly selectively encrypted. Our work can help removing possible barriers to the adoption of cloud-based services and enable their wider and confident adoption.

Acknowledgments

This work was supported in part by the EC under projects Chips JU EdgeAI (101097300) and GLACIATION (101070141), by the Italian MUR under PRIN project POLAR (2022LA8XBH), and by project SERICS (PE00000014) under the MUR NRRP funded by the EU - NGEU. Views and opinions expressed are however those of the authors only and do not necessarily reflect those of the European Union or the Italian MUR. Neither the European Union nor the Italian MUR can be held responsible for them.

References

[1] S. De Capitani di Vimercati, S. Foresti, S. Paraboschi, and P. Samarati, "Enforcing corporate governance's internal controls and audit in the cloud," in *Proc. of IEEE CLOUD*, Beijing, China, Oct. 2020.

[2] S. De Capitani di Vimercati, S. Foresti, and P. Samarati, "Protecting data and queries in cloud-based scenarios," *SN Computer Science*, vol. 4, no. 5, Sep. 2023.

[3] S. De Capitani di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, and P. Samarati, "Encryption policies for regulating access to outsourced data," *ACM TODS*, vol. 35, no. 2, pp. 12:1–12:46, Apr. 2010.

[4] M. Atallah, M. Blanton, N. Fazio, and K. Frikken, "Dynamic and efficient key management for access hierarchies," *ACM TISSEC*, vol. 12, no. 3, pp. 18:1–18:43, Jan. 2009.

[5] C. Ardagna, R. Asal, E. Damiani, T. Dimitrakos, N. El Ioini, and C. Pahl, "Certification-based cloud adaptation," *IEEE TSC*, vol. 14, no. 1, pp. 82–96, Jan.-Feb. 2021.

[6] M. Anisetti, C. Ardagna, E. Damiani, and F. Gaudenzi, "A semi-automatic and trustworthy scheme for continuous cloud service certification," *IEEE TSC*, vol. 13, no. 1, pp. 30–43, Jan.-Feb. 2020.

[7] R. Jhawar and V. Piuri, "Fault tolerance and resilience in cloud computing environments," in *Computer and Information Security Handbook, 2nd Edition*, J. Vacca, Ed. Morgan Kaufmann, 2013, pp. 125–141.

[8] P. Samarati and S. De Capitani di Vimercati, "Cloud security: Issues and concerns," in *Encyclopedia on Cloud Computing*, S. Murugesan and I. Bojanova, Eds. Wiley, 2015.

[9] H. Hacigümüş, B. Iyer, S. Mehrotra, and C. Li, "Executing SQL over encrypted data in the database-service-provider model," in *Proc. of SIGMOD*, Madison, WI, USA, Jun. 2002.

[10] X. Ding, Z. Wang, P. Zhou, K.-K. R. Choo, and H. Jin, "Efficient and privacy-preserving multi-party skyline queries over encrypted data," *IEEE TIFS*, vol. 16, pp. 4589–4604, Aug. 2021.

[11] G. Poh, J. Chin, W. Yau, K.-K. R. Choo, and M. Mohamad, "Searchable symmetric encryption: Designs and challenges," *ACM CSUR*, vol. 50, no. 3, pp. 1–37, May 2017.

[12] F. Li, J. Ma, Y. Miao, X. Liu, J. Ning, and R. H. Deng, "A survey on searchable symmetric encryption," *ACM CSUR*, vol. 56, no. 5, May 2024.

[13] S. De Capitani di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, R. Sassi, and P. Samarati, "Sentinels and twins: Effective integrity assessment for distributed computation," *IEEE TPDS*, vol. 34, no. 1, pp. 108–122, Jan. 2023.

[14] B. Zhang, B. Dong, and W. Wang, "Integrity authentication for sql query evaluation on outsourced databases: A survey," *IEEE TKDE*, vol. 33, no. 4, pp. 1601–1618, Apr. 2021.

[15] Y. Zhang, R. Deng, S. Xu, J. Sun, Q. Li, and D. Zheng, "Attribute-based encryption for cloud computing access control: A survey," *ACM CSUR*, vol. 53, no. 4, pp. 1–41, July 2021.

[16] S. Xu, J. Ning, X. Huang, Y. Li, and G. Xu, "Untouchable once revoking: A practical and secure dynamic EHR sharing system via cloud," *IEEE TDSC*, vol. 19, no. 6, pp. 3759–3772, Nov./Dec. 2022.

[17] S. Ruj, M. Stojmenovic, and A. Nayak, "Privacy preserving access control with authentication for securing data in clouds," in *Proc. of CCGrid*, Ottawa, Canada, May 2012.

[18] Q. Huang, Y. Yang, and M. Shen, "Secure and efficient data collaboration with hierarchical attribute-based encryption in cloud computing," *Future Generation Computer Systems*, vol. 72, July 2017.

[19] P. Perazzo, F. Righetti, M. La Manna, and C. Vallati, "Performance evaluation of attribute-based encryption on constrained IoT devices," *Comput. Commun.*, vol. 170, pp. 151–163, Mar. 2021.

[20] S. Yu, C. Wang, K. Ren, and W. Lou, "Attribute based data sharing with attribute revocation," in *Proc. of ASIACCS*, Beijing, China, Apr. 2010.

[21] D. Ghopur, J. Ma, X. Ma, J. Hao, T. Jiang, and X. Wang, "Puncturable key-policy attribute-based encryption scheme for efficient user revocation," *IEEE TSC*, vol. 16, no. 6, pp. 3999–4011, Nov.-Dec. 2023.

[22] E. Bacis, S. De Capitani di Vimercati, S. Foresti, S. Paraboschi, M. Rosa, and P. Samarati, "Mix&Slice for efficient access revocation on outsourced data," *IEEE TDSC*, vol. 21, no. 3, pp. 1390–1405, May 2023.

[23] S. De Capitani di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, and P. Samarati, "Over-encryption: Management of access control evolution on outsourced data," in *Proc. of VLDB*, Vienna, Austria, Sep. 2007.

[24] G. Miklau and D. Suciu, "Controlling access to published data using cryptography," in *Proc. of VLDB*, Berlin, Germany, Sep. 2003.

[25] R. Cimorelli Belfiore, A. De Santis, A. L. Ferrara, and B. Masucci, "Hierarchical key assignment schemes with key rotation," in *Proc. of SACMAT*, San Antonio, TX, USA, June 2024.

[26] S. De Capitani di Vimercati, S. Foresti, S. Jajodia, G. Livraga, S. Paraboschi, and P. Samarati, "Enforcing dynamic write privileges in data outsourcing," *COSE*, vol. 39, pp. 47–63, Nov. 2013.

[27] J. Leukel, S. Kirn, and T. Schlegel, "Supply chain as a service: A cloud perspective on supply chain systems," *IEEE Systems Journal*, vol. 5, no. 1, pp. 16–27, Jan. 2011.

[28] H. Zhang, T. Nakamura, and K. Sakurai, "Security and trust issues on digital supply chain," in *Proc. of DASC/PiCom/CBDCom/CyberSciTech*, Fukuoka, Japan, Aug. 2019.

[29] A. Tueno, F. Kerschbaum, D. Bernau, and S. Foresti, "Selective access for supply chain management in the cloud," in *Proc. of SPC*, Las Vegas, NV, USA, Oct. 2017.

Sabrina De Capitani di Vimercati is a professor at the Università degli Studi di Milano, Italy. Her research interests are in data security and privacy. She has published more than 230 papers in journals, conference proceedings, and books. She has been a visiting researcher at SRI International, CA, USA, and George Mason University, VA, USA.
https://decapitani.di.unimi.it

Sara Foresti is a professor at the Università degli Studi di Milano, Italy. Her research interests are in data security and privacy. She has published more than 120 papers in journals, conference proceedings, and books. She has been a visiting researcher at George Mason University, VA, USA. She chairs the IFIP WG 11.3 on Data and Applications Security and Privacy.
https://foresti.di.unimi.it

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

Pierangela Samarati is a professor at the Università degli Studi di Milano, Italy. Her main research interests are in data protection, security, and privacy. She has published more than 300 papers in journals, conference proceedings, and books. She has been a visiting researcher at Stanford University, CA, USA, SRI International, CA, USA, and George Mason University, VA, USA. She is a Fellow of ACM, IEEE, and IFIP.
https://samarati.di.unimi.it