Enforcing Corporate Governance's Internal Controls and Audit in the Cloud

Sabrina De Capitani di Vimercati*, Sara Foresti*, Stefano Paraboschi†, Pierangela Samarati*
* *Università degli Studi di Milano, Italy – Email:* name.surname@*unimi.it*
† *Università degli Studi di Bergamo, Italy – Email: parabosc@unibg.it*

***Abstract*—More and more organizations are today using the cloud for their business as a quite 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 functions that, already in-house, may result sensitive or security critical, and whose enforcement in the cloud requires then 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 is based on the use of selective encryption and of tags to provide a level of self-protection to data and for enabling only authorized parties to access data and perform operations on them, providing privacy and integrity guarantees, as well as accountability and non-repudiation.**

***Keywords*-Cloud-based services; outsourcing; internal controls and audit process; selective encryption**

I. 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. As an example, the European Corporate Governance Codes Networks (www.ecgcn.org) has the goal to share experience on issues related to corporate governance. These national Corporate Governance Codes contain many principles focusing on different aspects, such as the model of corporate governance, and relations with shareholders; an important role is played by 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 successful realization can contribute to the improvement of the quality of corporate governance and management. ICA functions can be performed in different 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, and 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.

The adoption of cloud-based solutions for supporting ICA functions requires to ensure confidentiality and integrity of data and of controls on them, involving different challenges. First, data records subject to internal controls and audit may need to remain confidential to the cloud provider itself. Second, access to data and execution of internal controls and audit should be possible only for authorized parties. Third, parties should be accountable for their actions. Fourth, parties involved in the control should be able to assess integrity of information (data and results of control) on which they operate. Although several proposals have been designed for the protection of data in the cloud (e.g., [1]), none of them can be directly applied to correctly enforce the ICA of corporate governance.

In this paper, we consider the problem of securely relying on the cloud for supporting ICA functions and address all the challenges above. The remainder of this paper is organized as follows. Section II presents the scenario, introducing the parties involved in the ICA functions and their execution. Section III illustrates our solution for providing protection to data in the cloud and enforcing access control on them so that only authorized parties can access them. Section IV enriches such solution with capability of secure support of all ICA functions ensuring their execution only to authorized parties, with confidentiality and integrity guarantees. Section V provides the algorithm implementing the proposed solutions. Section VI discusses related work. Finally, Section VII concludes the paper.

II. Scenario and problem definition

We consider an organization composed of different operating *units* whose *employees* can 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

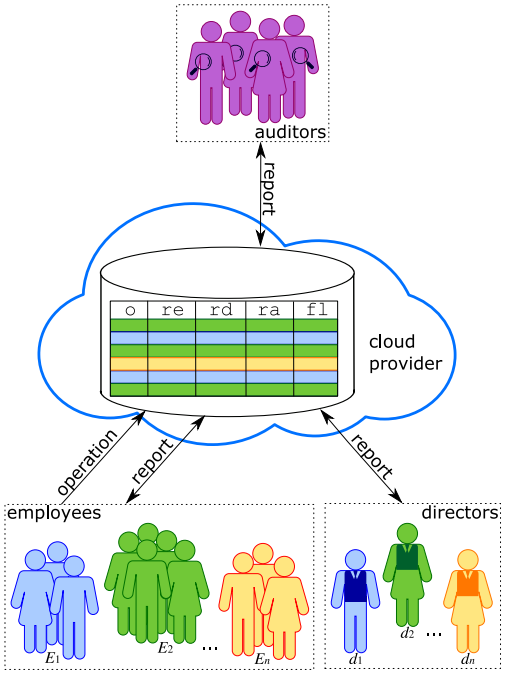

Figure 1. ICA process at an external cloud provider

each director is responsible for one unit only. Formally, we denote the set of units of the organization as U, and, for each unit $u_i \in U$, we denote its employees as E_i and its director as d_i. Each unit u_i stores the information of operations in a log, modeled as a set O_i of operation records. Each record stores the information of the corresponding operation, which includes a unique operation identifier `id` and the identifier of the unit `u` where the operation has been processed. We use the traditional dot notation to refer to a specific attribute of an operation record (e.g., $o.\texttt{id}$ denotes the identifier of operation o). For simplicity, in the following, we will use the term *operation* to refer interchangeably to the operation and its record. Notation D, $\mathcal{E}$, and $\mathcal{O}$ is used to refer to the set of directors, employees, and operations of all units, respectively (i.e., $D = \{d_1, \ldots, d_n\}$, $\mathcal{E} = E_1 \cup \ldots \cup E_n$, and $\mathcal{O} = O_1 \cup \ldots \cup O_n$). Note that the directors of units are disjoint from the set of employees, that is, $D \cap \mathcal{E} = \emptyset$. Given an employee $e \in \mathcal{E}$, *unit*(e) denotes the unit for which the employee works. Analogously, given a director $d \in D$, *unit*(d) denotes the unit directed by d.

All operations must undergo an *internal controls and audit* (ICA) process to ensure that they are legitimate. This process involves different subjects, that is, the employees of the organizations, the directors, and a set $A = \{a_1, \ldots, a_w\}$ of independent auditors appointed by the organization. Each subject is responsible of different checks on the operations. Given an operation, a first level of control is executed by an employee of the unit where the operation has been processed. The result of this check is a report that is stored in association with the operation itself (as attribute $o.\texttt{re}$). The second level of control is performed by the director of the unit. The director must check both the operation and the report generated during the first level of control. Also in this case, at the end of the check, a second report is created (as attribute $o.\texttt{rd}$). Finally, the third level of control is performed by an *independent auditor* who checks the operation and the reports created in the previous steps. The independent auditor produces a final report for the operation (as attribute $o.\texttt{ra}$) and marks the internal controls and audit process as concluded (attribute $o.\texttt{fl}$ set to TRUE).

Example 2.1: We consider, as a running example, an organization with two units U={X, Y}, where unit X has three employees E_x={x_1, x_2, x_3} and is directed by d_x, and unit Y has two employees E_y={y_1, y_2} and is directed by d_y. The set of operations processed by units X and Y are O_x and O_y, respectively.

The management of operations and of the corresponding internal controls and audit operate in compliance with a regulation established by the organization, according to which: *i)* information related to an operation is accessible to all employees and the director of the unit where the operation has been processed; and *ii)* information related to all operations of all the units is accessible to all independent auditors. These two requirements naturally translate into an access control policy regulating which subject (employee, director, auditor) should be authorized to access the operations. Given the set $\mathcal{S}$=$\mathcal{E}\cup D\cup A$ of subjects in the system, the access control policy of the organization can be formally defined as follows.

Definition 2.1 (Access control policy): Let $\mathcal{S}$=$\mathcal{E}\cup D\cup A$ be the set of subjects, U be the set of units, and $\mathcal{O}$ be the set of operations. An *access control policy* $\mathcal{P}$ regulating access to operations in $\mathcal{O}$ is a set of permissions of the form $\langle s,o\rangle$, with $s\in\mathcal{S}$, $o\in\mathcal{O}$, such that only the following permissions are included:

1) $\forall s \in \mathcal{E} \cup D$, $\forall o \in \mathcal{O} : \langle s,o\rangle \in \mathcal{P}$ iff $o.\texttt{u}$=*unit*(s);
2) $\forall a \in A$, $\forall o \in \mathcal{O} : \langle a,o\rangle \in \mathcal{P}$.

Each pair $\langle s, o\rangle$ in $\mathcal{P}$ states that s can access o.

Due to the huge number of operations that are processed on a daily basis, the operation records as well as the corresponding reports generated during the internal controls and audit process are stored and managed in the cloud (see Figure 1). Our goal is therefore the design of a solution that supports the external storage of the operation records while guaranteeing the confidentiality and integrity of their content as well as the integrity and non repudiation of ICA reports. Indeed, the cloud provider, which is not under the direct control of the organization, is assumed to be *honest-but-curious* meaning that it is trustworthy to properly manage the ICA process but it may not be trusted to read the operations content and reports. Figure 2 summarizes the notation used in the paper. In the following, we first describe how to

U	set of units
$\mathcal{E}$	set of employees of the organization
D	set of directors of the organization
A	set of independent auditors
$\mathcal{S}$	set of subjects ($\mathcal{E} \cup D \cup A$)
$\mathcal{O}$	set of operations processed at the organization
E_i	set of employees of unit u_i
O_i	set of operations processed at unit u_i
u_i	unit i
d_i	director of unit u_i
unit(e)	unit of employee $e \in \mathcal{E}$
unit(d)	unit of director $d \in D$
o.id	identifier of operation o
o.u	unit where operation o has been processed
o.re	employee report of operation o
o.rd	director report of operation o
o.ra	auditor report of operation o
o.fl	flag of status of operation o
$\mathcal{P}$	access control policy of the organization

Figure 2. Notation used in the paper

enforce the access control policy of the organization using selective encryption (Section III) and then shows how to enforce the ICA process (Section IV and Section V).

III. Storage and access control

Before moving the storage of the operations to the cloud, the record of each operation is encrypted to guarantee its confidentiality with respect to the cloud provider. Our solution enforces the access control policy of the organization over the encrypted operations externally stored leveraging the *selective encryption* technique (e.g., [2]). Basically, we encrypt different operations with different encryption keys and ensure that each subject can decrypt all and only the operations she is authorized to access. To do so without requiring the distribution of multiple keys to subjects, we organize keys in a hierarchical structure enabling a subject to derive from a single key (the one assigned to the subject) all, and only, the keys used for encrypting the operations that the subject is authorized to access (e.g., [3]). Each subject s with key k_s can then derive another key k through a public *token* computed as t=$k \oplus h(k_s,l)$, where h is a deterministic cryptographic function, $\oplus$ is the bitwise xor operator, and l is a publicly available label associated with key k. The derivation process can be direct or indirect, through a chain of tokens (i.e., key k can in turn be used to derive another key k'). Intuitively, key derivation permits to assign a single key to each subject (employee, director, auditor), from which she can derive the encryption keys used to protect the operations that she is authorized to access. The access control policy $\mathcal{P}$ can then be translated into an *equivalent encryption policy* dictating the key used to encrypt each operation, the key released to each subject, and key derivation via tokens. Basically, each employee, director, and auditor has her own key. Also, since all operations processed at a specific unit can be accessed by the same set of subjects (i.e., the employees and the director of the unit as well as all auditors), they can be all encrypted with the same key. We can therefore associate a key with each unit and use it for encrypting all the operations processed at the unit. The employees and director of each unit must then be able to derive, from their own key, the key of the unit where they work. The auditors must instead be able to derive, from their own key, the keys of all the units. Formally, an encryption policy equivalent to the access control policy $\mathcal{P}$ of the organization is defined as follows.

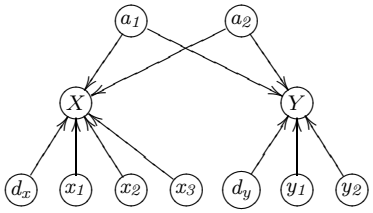

Figure 3. Key derivation structure for Example 3.1

Definition 3.1 (Equivalent encryption policy): Let $\mathcal{P}$ be the access control policy over the set U of units and the set $\mathcal{S}$=$\mathcal{E} \cup D \cup A$ of subjects, and $\mathcal{O}$ be the set of operations. An *encryption policy* Π *equivalent* to $\mathcal{P}$ is a 7-uple $\langle \mathcal{S},\mathcal{O},\mathcal{K},\mathcal{L},\phi_{\mathcal{S}},\phi_{\mathcal{O}},T \rangle$ where:

1) ($\mathcal{K}$,$\mathcal{L}$) is the set of keys and corresponding labels such that $\mathcal{K}$=$K_{\mathcal{S}} \cup K_U$, with $K_{\mathcal{S}}$=$\{k_s | \ s \in \mathcal{S}\}$ and K_U=$\{k_u | \ u \in U\}$;
2) $\phi_{\mathcal{S}} : \mathcal{S} \rightarrow K_{\mathcal{S}}$ is an injective function assigning a key and corresponding label to each subject;
3) $\phi_{\mathcal{O}} : \mathcal{O} \rightarrow K_U$ is a function assigning a key and corresponding label to each operation such that $\forall o', o'' \in \mathcal{O}$ with o'.u=o''.u, $\phi_{\mathcal{O}}(o')$=$\phi_{\mathcal{O}}(o'')$;
4) T is the set of tokens defined over $\mathcal{K}$ and $\mathcal{L}$ such that: $\forall \langle s, o \rangle \in \mathcal{P}$, $\phi_{\mathcal{S}}(s) \xrightarrow{T} \phi_{\mathcal{O}}(o)$; $\forall k_w \xrightarrow{T} k_v$, with $k_w \in K_{\mathcal{S}}, k_v \in K_U$, $\forall o \in {\phi_{\mathcal{O}}}^{-1}(k_v)$, $\langle {\phi_{\mathcal{S}}}^{-1}(k_w), o \rangle \in \mathcal{P}$.

Notation $k' \xrightarrow{T} k''$ indicates that key k'' is derivable from key k' through a (sequence of) token(s) in T. The above definition of equivalent encryption policy says that: there is a key for each subject and each unit, and all operations of the same unit u are encrypted with the same key k_u (1-3), and each subject s is associated with a key $\phi_{\mathcal{S}}(s)$ from which she can derive all and only the keys of the operations that s is authorized to access according to policy $\mathcal{P}$ (4). Note that the definition does not say anything about how to create the tokens to guarantee policy equivalence. We can therefore adopt different strategies. Our solution consists in creating, for each unit $u \in U$, a token enabling the direct derivation of k_u from the key of each employee and director of u and from the key of each auditor. Each operation $o \in \mathcal{O}$ is then encrypted with key k_u, thus ensuring the correct enforcement of conditions 3 and 4 of Definition 3.1.

Example 3.1: Figure 3 graphically illustrates the key derivation structure of our running example, obtained ac-

cording to the above-mentioned approach. Each key is represented by a node and each token $t_{i,j}$ enabling the derivation of k_j from k_i is represented by an edge (k_i,k_j). For simplicity, in the figure we denote each key k_i with the subject/unit i to which it refers.

Note that each access by an authorized subject to an outsourced operation record requires a search across the set T of tokens (and labels $\mathcal{L}$) that are publicly available and stored on the cloud provider.

IV. ICA PROCESS IN THE CLOUD

Considering an operation o processed at unit u, the ICA process applied by our reference organization satisfies some requirements related to the management of the reports created after each ICA process phase. Such requirements can be summarized as follows.

R1) *Report generation.* The report of each ICA phase can be *generated* only by authorized subjects, namely:
R1.1) the employee report `re` can be prepared only by an employee e of unit u (i.e., *unit*(e)=u);
R1.2) the director report `rd` can be prepared only by the director d of unit u (i.e., *unit*(d)=u); and
R1.3) the audit report `ra` can be prepared only by an auditor $a \in A$.

R2) *Report update.* The report of each ICA phase can be *modified*:
R2.1) only by the subject who generated it; and
R2.2) only until the beginning of the subsequent phase.

R3) *Report integrity and accountability.* The report of each ICA phase should not be *tampered* with and *cannot be repudiated* by the subject who generated it, who is also responsible for its content.

In this section, we present our solution for the enforcement of these requirements when the operation records are encrypted and stored at an external cloud provider where no trusted monitor can directly enforce them.

A. Report generation

When moving the storage of operation records to the cloud, the enforcement of requirement R1 is complicated by the absence of a trusted party to whom controls can be delegated. Indeed, the cloud provider is not trusted for authentication and policy enforcement. The idea is then to design a solution that, in cooperation with the cloud provider, exploits selective encryption for the enforcement of requirement R1. Intuitively, our solution consists in associating a secret, encrypted using selective encryption, with each report to regulate (write) access to it. The secret is a value generated together with the operation and encrypted with a key known only to the subjects authorized to generate the report and to the cloud provider. The encrypted secret is called *tag* and the cloud provider will accept a write/create operation on a report related to a given operation when the requesting subject shows ability to decrypt the corresponding tag. Formally, a tag is defined as follows.

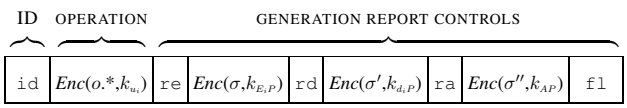

Figure 4. Structure of outsourced operation record

Definition 4.1 (Tag): A *tag* is the result of the encryption *Enc*(σ,k_{SP}) of a secret σ with a key k_{SP} known to the subjects in $S \subseteq \mathcal{E} \cup D \cup A$ and to the cloud provider P.

The secret σ used to compute a tag can be a randomly generated number. The proper enforcement of requirement R1 requires each operation to be associated with three different tags, one for each report, that must be encrypted with different keys shared by the cloud provider and the subjects authorized to generate the report. Formally, each operation o of unit u_i is associated with three tags computed as follows.

- `tag_e`=*Enc*(σ,k_{E_iP}), with k_{E_iP} a key known only to the employees E_i of unit u_i and to the cloud provider P, and σ the secret needed to create/write report `re` (R1.1).
- `tag_d`=*Enc*(σ',k_{d_iP}), with k_{d_iP} a key known only to the director d_i of unit u_i and to the cloud provider P, and σ' the secret needed to create/write report `rd` (R1.2).
- `tag_a`=*Enc*(σ'',k_{AP}), with k_{AP} a key known to the auditors A and to the cloud provider P, and σ'' the secret needed to create/write report `ra` (R1.3).

Figure 4 shows the structure of the outsourced records. As an example, operation o_x processed at unit X is associated with tags `tag_e`=*Enc*(σ,k_{E_XP}), `tag_d`=*Enc*(σ',k_{d_XP}), and `tag_a`=*Enc*(σ'',k_{AP}) enabling the employees of unit X, its director d_x, and all auditors to create/write report `re`, `rd`, and `ra`, respectively. Indeed, the cloud provider will permit to write report `re` (`rd` and `ra`, resp.) only to subjects demonstrating their ability to correctly decrypt `tag_e` (`tag_d` and `tag_a`, resp.). Operatively, a subject (i.e., an employee e, director d, and auditor a, resp.) before performing a check on an operation has to first download the appropriate tag from the cloud provider (i.e., `tag_e`, `tag_d`, or `tag_a`, resp.), decrypt it, and send the retrieved secret to the cloud provider. The cloud provider allows the subject to write the report only if the communicated secret corresponds to the secret that the provider retrieves decrypting the same tag (i.e., `tag_e`, `tag_d`, and `tag_a`, resp.).

The key derivation structure illustrated in Section III is then extended with the keys necessary for the management of tags. In particular, the structure is extended by adding a node for the key k_P of the cloud provider and a node for each key k_{SP} shared by a subset $S \subseteq \mathcal{S}$ of subjects and the

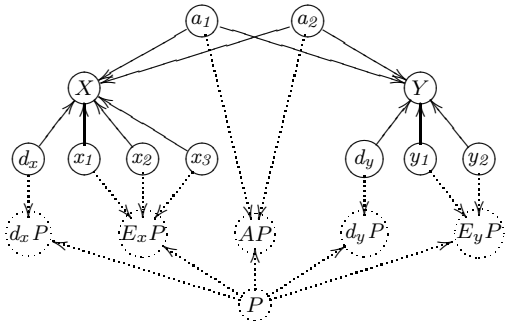

Figure 5. Extended key derivation structure for our running example

cloud provider. The cloud provider, as well as the authorized subjects, can derive these keys through a direct token from their own key. For instance, each employee $e \in E_i$ working at unit u_i can derive key k_{E_iP} through a token from key k_e to key k_{E_iP}.

Example 4.1: Consider our running example and the key derivation structure in Figure 3. Figure 5 illustrates the extended key derivation structure, where nodes with a dotted circle represent keys shared with the cloud provider, and dotted edges represent the tokens added to the original key derivation structure for allowing authorized subjects and the cloud provider to derive the keys necessary to decrypt tags.

B. Report update

Requirement R2 regulates who can update a report and until when. As an example, consider our running example and suppose that employee x_1 checks operation o_x and then generates report o_x.re. According to R2, only x_1 can modify o_x.re until director d_x begins the second level of control. As discussed in Section IV-A, tags represent an effective solution for regulating, in cooperation with the cloud provider, who can write reports. We then leverage tags to enforce also requirement R2 as described in the following.

Restrict updates (R2.1). Since a report can be generated/modified only by the subjects who can prove the knowledge of the corresponding secret (σ) in the tag (i.e., the subjects who can correctly decrypt the tag), updates to the report can be prevented by changing the secret and therefore the tag. In this way, only presenting the new secret a subject can modify the corresponding report. Note that the secret must be changed and encrypted with a key that only the subjects authorized to modify the report (and the cloud provider) know, to avoid that previously authorized subjects can present the old secret and obtain the access. We apply this idea to restrict updates to the employee and auditor reports (the director report can already be updated only by the director). Whenever an authorized subject $s \in \mathcal{E} \cup A$ (i.e., an employee or auditor) starts the analysis of an operation o, the subject has to: *1)* generate a new secret σ'; *2)* create a new tag obtained encrypting the new secret with key k_{sP} shared with the cloud provider (*Enc*(σ', k_{sP})); *3)* overwrite the old tag (i.e., o.tag_e or o.tag_a, depending on whether s is an employee or an auditor) with the new one. Since the new secret σ' is encrypted with key k_{sP}, only subject s can prove to the cloud provider the knowledge of σ', thus blocking the possible attempts by previously authorized subjects (i.e., the employees of the same unit as s and all the auditors) to modify the report. The new keys needed to encrypt the new secrets (i.e., $\forall s \in \mathcal{E} \cup A$, key k_{sP}) are derivable by the authorized subjects (i.e., each subject $s \in \mathcal{E} \cup A$) and by the cloud provider through tokens added to the key derivation structure and that start from the key of the authorized subjects and from the key of the provider.

	tag_e	tag_d	tag_a	fl
ICA process	*Enc*(σ_1,k_{E_xP})	*Enc*(σ_2,k_{d_xP})	*Enc*(σ_3,k_{AP})	FALSE
1st start, x_1	*Enc*(σ_4,k_{x_1P})	*Enc*(σ_2,k_{d_xP})	*Enc*(σ_3,k_{AP})	FALSE
2nd start, d_x	dummy	*Enc*(σ_2,k_{d_xP})	*Enc*(σ_3,k_{AP})	FALSE
3rd start, a_1	dummy	dummy	*Enc*(σ_5,k_{a_1P})	FALSE
3rd end, a_1	dummy	dummy	dummy	TRUE

Figure 6. An example of evolution of tags for operation o_x

Prevent updates (R2.2). A subject who has generated a report can modify it until the subsequent control phase, that is, when a director or auditor starts a control, the subject who has created the report in the previous phase (i.e., an employee or director, resp.) should not be able to update it anymore. To prevent such updates, it is sufficient to substitute the tag of the previous phase with a dummy content that cannot be decrypted. In this way, since the tag cannot be decrypted, the provider as well as any subject cannot retrieve the secret and then nobody is authorized to update the report. Note that also when an auditor terminates her control, the corresponding tag tag_a is overwritten with a dummy content. Furthermore, the auditor sets attribute fl to TRUE. When attribute fl is TRUE the ICA process on the corresponding operation is considered concluded and the cloud provider does not accept any write operation on any reports.

Example 4.2: Consider operation o_x of unit X of our running example. Figure 6 illustrates the evolution of tag_e, tag_d, and tag_a to satisfy requirement R2, assuming that x_1, d_x, and a_1 perform the first, second, and third level of control, respectively. In the figure, updates to tags are in black, while unchanged tags are in gray. Note that, while any employee in E_x can start the first level of control, since they can all retrieve σ_1 from tag_e, when x_1 starts the first level of control, x_2 and x_3 cannot modify re since they cannot decrypt tag_e (they do not know k_{x_1P}) and σ_1 is no more accepted by the provider. Similarly, when tag_e is set to dummy by the director, also x_1 cannot modify re. Figure 7 illustrates the key derivation structure in Figure 5, extended with the keys necessary to manage tags for the enforcement of requirement R2.

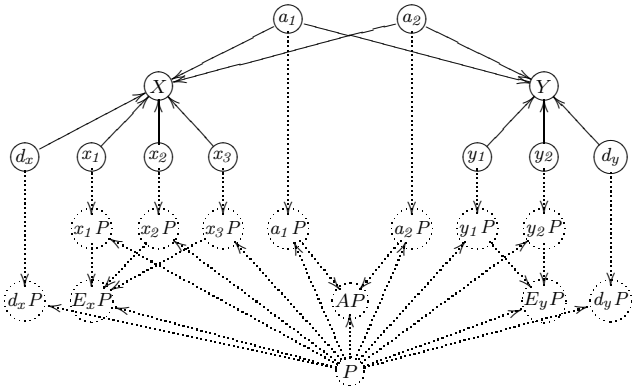

Figure 7. Extended key derivation structure for our running example, with keys for tags evolution

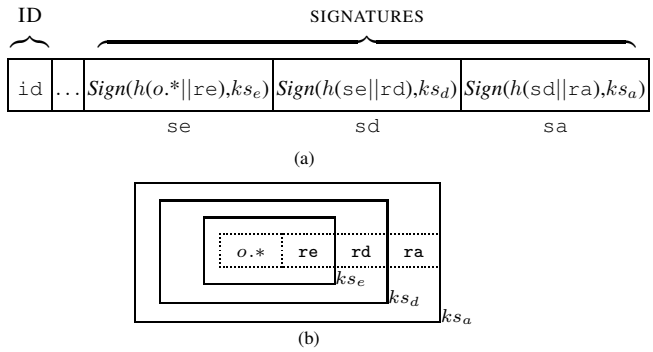

Figure 8. Signatures in the outsourced operation record (a) and their layered structure (b)

C. Report integrity and accountability

Requirement R3 has the objective to verify that all subjects involved in the ICA process as well as the cloud provider behave properly. In particular, it aims at ensuring that modifications to reports have been produced only by authorized subjects. If all subjects behave as discussed in the previous sections and the cloud provider performs the correct control on tags, the integrity of operations and reports is automatically guaranteed. However, since unauthorized updates to reports cannot be prevented, we adopt a signature-based approach that allows the detection of possible tampering of operations and reports. Each subject has a pair $\langle ks,kp\rangle$ of secret and public keys and uses her secret key to sign the reports she generates. This guarantees that any tampering to an operation or to its reports can be immediately detected by any subject knowing the public key kp of the subject who generated the report. Also, by signing a report, the subject who generated the report cannot repudiate it. Concretely, given a hash function h, each operation record is associated with the following three signatures, one for each subject involved in the three controls of the ICA process.

- `se`=*Sign*(h(*o*.*||`re`), ks_e), with *o*.* the set of all attributes in o and e the employee who generated report `re`.
- `sd`=*Sign*(h(`se`||`rd`), ks_d), with d the director who generated report `rd`.
- `sa`=*Sign*(h(`sd`||`ra`), ks_a), with a the auditor who generated report `ra`.

Figure 8(a) illustrates the structure of the portion of the outsourced records that includes the three signatures. Each signature guarantees the integrity of the corresponding report and makes the subject who produced the report accountable for its content. Since the public keys used to generate `se`, `sd`, and `sa` are publicly available, all subjects in the system and the cloud provider can verify the signatures and detect tampering of reports and operations. Note that the signature produced at the end of a phase of the ICA process is applied on the digest of the concatenation of the report produced in that phase and the signature computed at the end of the previous phase (with the operation content, for the employee report). By signing the signature computed in the previous phase, the subject implicitly confirms that she has verified such signature before starting her control. Figure 8(b) shows the layered organization of the signatures. Note that this layered structure, which starts with the signature of an employee computed over the operation content together with the employee report, ensures the correct association of each operation with its reports. For instance, if a malicious subject copies the content of o_i.`re` into o_j.`re`, signature o_j.`se` would signal the inconsistency of the report content.

Example 4.3: With respect to Example 4.2, when employee x_1 has generated report `re`, she signs o_x concatenated with `re`, using her secret key ks_{x_1}. Director d_x can then verify `se` with key kp_{x_1}, generate report `rd`, and sign the concatenation of `se` and `rd` with her secret key ks_{d_x}. Similarly, auditor a_1 can verify `sd` with kp_{d_x}. When the auditor generates report `ra`, she signs the concatenation of `sd` and `ra` with her secret key ks_{a_1}.

V. Operation generation and control

We describe the functions (Figure 9) implementing the solutions described in the previous sections for the storage of operations and the enforcement of the ICA process.

Generation. Each operation is generated by an employee working at a unit $u_i \in U$. The operation record is encrypted with the key k_{u_i} of the unit (line 1). The encrypted record is also associated with a unique identifier (line 2) and with the tags regulating the generation of reports (lines 3–6). While tag `tag_e` can be generated by any employee of the unit, `tag_d` and `tag_a` can be generated only by the director of the unit and by auditors, respectively. To avoid requesting continuous online presence of directors and auditors, we assume that the director of the unit and auditors generate a pool of tags in advance (sets $\texttt{DirTags}_{u_i}$ and `AuditTags`, resp.). When an operation is generated, it is associated with tags extracted from these sets. Reports and signatures for

OPERATION MANAGEMENT

Generate_Encrypted_Operation(o) @ u_i
1: o^k.enc := *Enc*(o.*, k_{u_i})
2: o^k.id := generate an identifier
3: randomly generate σ_1
4: o^k.tag_e := *Enc*(σ_1, k_{E_iP})
5: o^k.tag_d := **ExtractTag**($\mathtt{DirTags}_{u_i}$) /* tags for director of u_i */
6: o^k.tag_a := **ExtractTag**(AuditTags) /* tags for auditors */
7: o^k.re := o^k.rd := o^k.ra := NULL
8: o^k.se := o^k.sd := o^k.sa := NULL
9: o^k.fl := FALSE
10: upload o^k on the cloud provider

First_Level_Control(id) @ $e \in E_i$
1: o^k := download o with identifier id from the provider
2: σ_1 := *Dec*(o^k.tag_e, k_{E_iP})
3: **if** **Verify**(id, 1, σ_1, E_i)=TRUE **then**
4: randomly generate σ_4
5: o^k.tag_e := *Enc*(σ_4, k_{eP})
6: upload o^k on the cloud provider
7: o := *Dec*(o^k.enc, k_{u_i})
8: *report* := generate the report for o
9: o^k.re := *Enc*(*report*, k_{u_i})
10: o^k.se := *Sign*(h(o.*||*report*), ks_e)
11: upload o^k on the cloud provider

Second_Level_Control(id) @ d_i
1: o^k := download o with identifier id from the provider
2: σ_2 := *Dec*(o^k.tag_d, k_{d_iP})
3: **if** **Verify**(id, 2, σ_2, $\{d_i\}$)=TRUE **then**
4: o^k.tag_e := dummy
5: upload o^k on the cloud provider
6: o := *Dec*(o^k.enc, k_{u_i})
7: o.re := *Dec*(o^k.re, k_{u_i})
8: **if** **Verify_Signature**(o^k.se, h(o.*||o.re), kp_e) **then**
9: *report* := generate the report for o
10: **else** *report* := not passed
11: o^k.rd := *Enc*(*report*, k_{u_i})
12: o^k.sd := *Sign*(h(o^k.se||*report*), ks_{d_i})
13: upload o^k on the cloud provider

Third_Level_Control(id) @ a
1: o^k := download o with identifier id from the provider
2: σ_3 := *Dec*(o^k.tag_a, k_{AP})
3: **if** **Verify**(id, 3, σ_3, A)=TRUE **then**
4: randomly generate σ_5
5: o^k.tag_a := *Enc*(σ_5, k_{aP})
6: o^k.tag_d := dummy
7: upload o^k on the cloud provider
8: o := *Dec*(o^k.enc, k_{u_i})
9: o.re := *Dec*(o^k.re, k_{u_i})
10: o.rd := *Dec*(o^k.rd, k_{u_i})
11: **if** **Verify_Signature**(o^k.sd, h(o^k.se||o.rd), kp_{d_i}) **then**
12: *report* := generate the report for o
13: **else** *report* := not passed
14: o^k.ra := *Enc*(*report*, k_{u_i})
15: o^k.sa := *Sign*(h(o^k.sd||*report*), ks_a)
16: o^k.tag_a := dummy
17: o^k.fl := TRUE
18: upload o^k on the cloud provider

Verify(id, *phase*, σ, X) @ P
1: let o^k be the operation with o^k.id=id
2: **case** *phase* **of**
3: 1: *proof* := *Dec*(o^k.tag_e, k_{XP})
4: 2: *proof* := *Dec*(o^k.tag_d, k_{XP})
5: 3: *proof* := *Dec*(o^k.tag_a, k_{XP})
6: **if** σ=*proof* **then return**(TRUE)
7: **return**(FALSE)

Figure 9. Pseudocode of operation generation and ICA process

the generated operation are set to NULL, while fl is set to FALSE (lines 7–9). When all information about the operation has been set, the record is uploaded on the cloud provider (line 10).

First level control. When employee e performs the first control on operation o, she first needs to prove to the cloud provider her ability to decrypt tag_e (lines 1–3). The employee then overwrites tag_e with a tag that other employees cannot decrypt (lines 4–6). Note that the update to tag_e is immediately uploaded on the server. The employee then decrypts the operation record, performs the control, and generates the corresponding report (lines 7-8). She encrypts the report using the key of the unit k_{u_i} and computes signature se (lines 9–10). Finally, the operation record is uploaded on the cloud provider (line 11).

Second level control. When the unit director d_i needs to control operation o, she first decrypts tag_d to demonstrate to the cloud provider that she is entitled to generate the report (lines 1–3). The director then nullifies tag_e (lines 4–5). She decrypts the operation record and report re, verifies signature se, and performs the control (lines 6-10). The director encrypts her report with the key of the unit k_{u_i} and computes signature sd (lines 11–12). Finally, the operation record is uploaded on the cloud provider (line 13).

Third level control. When auditor a controls operation o, she first decrypts tag_a to demonstrate that she is authorized to perform the control (lines 1–3). The auditor then overwrites tag_a with a tag that other auditors cannot decrypt, and nullifies tag_d (lines 4–7). She decrypts the operation record and the reports of the first two controls, re and rd, and verifies signature sd (lines 8-11). The auditor generates her report, encrypts it with k_{u_i}, and computes sa (lines 12–15). The auditor finally nullifies tag_a, sets fl to TRUE (lines 16–17), and the operation record is uploaded on the cloud provider (line 18).

Note that when one of the controls of an operation fails, the record associated with the operation is not deleted. The unit is alerted and further controls are put in place at the organization level, generating a new report for the operation.

Example 5.1: Figure 10 illustrates the evolution of the record of operation o_x of our running example. The table has a column for each generation report control and signature attribute, and a row for each ICA phase. We assume that o_x is verified by employee x_1, director d_x, and auditor a_1.

VI. RELATED WORK

The problem of moving data storage and management to the cloud has been widely studied. Since cloud providers are not necessarily trusted, data confidentiality, integrity, and availability are possibly at risk when outsourcing data [1]. Several efforts have been then devoted to the definition of approaches enabling the effective and efficient evaluation of computations and queries directly over encrypted data

	re	tag_e	rd	tag_d	ra	tag_a	fl	se	sd	sa
ICA process	NULL	$Enc(\sigma_1, k_{E_xP})$	NULL	$Enc(\sigma_2, k_{d_xP})$	NULL	$Enc(\sigma_3, k_{AP})$	FALSE	NULL	NULL	NULL
1st start, x_1	NULL	$Enc(\sigma_4, k_{x_1P})$	NULL	$Enc(\sigma_2, k_{d_xP})$	NULL	$Enc(\sigma_3, k_{AP})$	FALSE	NULL	NULL	NULL
1st end, x_1	$Enc(\texttt{re}, k_X)$	$Enc(\sigma_4, k_{x_1P})$	NULL	$Enc(\sigma_2, k_{d_xP})$	NULL	$Enc(\sigma_3, k_{AP})$	FALSE	$Sign(h(o_x\|\|\texttt{re}), ks_{x_1})$	NULL	NULL
2nd start, d_x	$Enc(\texttt{re}, k_X)$	dummy	NULL	$Enc(\sigma_2, k_{d_xP})$	NULL	$Enc(\sigma_3, k_{AP})$	FALSE	$Sign(h(o_x\|\|\texttt{re}), ks_{x_1})$	NULL	NULL
2nd end, d_x	$Enc(\texttt{re}, k_X)$	dummy	$Enc(\texttt{rd}, k_X)$	$Enc(\sigma_2, k_{d_xP})$	NULL	$Enc(\sigma_3, k_{AP})$	FALSE	$Sign(h(o_x\|\|\texttt{re}), ks_{x_1})$	$Sign(h(\texttt{se}\|\|\texttt{rd}), ks_{d_x})$	NULL
3rd start, a_1	$Enc(\texttt{re}, k_X)$	dummy	$Enc(\texttt{rd}, k_X)$	dummy	NULL	$Enc(\sigma_5, k_{a_1P})$	FALSE	$Sign(h(o_x\|\|\texttt{re}), ks_{x_1})$	$Sign(h(\texttt{se}\|\|\texttt{rd}), ks_{d_x})$	NULL
3rd end, a_1	$Enc(\texttt{re}, k_X)$	dummy	$Enc(\texttt{rd}, k_X)$	dummy	$Enc(\texttt{ra}, k_X)$	dummy	TRUE	$Sign(h(o_x\|\|\texttt{re}), ks_{x_1})$	$Sign(h(\texttt{se}\|\|\texttt{rd}), ks_{d_x})$	$Sign(h(\texttt{sd}\|\|\texttt{ra}), ks_{a_1})$

Figure 10. An example of evolution of the tags, reports, and signatures for operation o_x

(e.g., [4], [5], [6]) and providing integrity of their results (e.g., [7], [8], [9]). For enforcing access control, researchers have investigated approaches that rely on attribute-based encryption based on public keys (e.g., [10], [11], [12], [13], [14]) and selective encryption based on hierarchically organized symmetric keys (e.g., [2], [15], [16]). Attribute-Based Encryption (ABE) regulates access to resources according to access control policies that are defined on attributes associated with every user's secret key and on an access structure associated with every resource (or vice versa). A user can access a resource when her attributes match the access structure associated with the resource (or vice versa). Selective encryption techniques translate the authorization policy regulating access to resources into an equivalent encryption policy. Each resource is then encrypted with a different key that only authorized users can derive from their own secret key. The management of policy updates is complicated when using both ABE and selective encryption and solutions for both techniques have been proposed (e.g., [17], [18]).

Most of the current solutions for access control enforcement in cloud scenarios mainly focus on the restriction of read operations over sensitive data. A few works have addressed the problem of enforcing restrictions on write operations, which are usually assumed to be an exclusive privilege of the data owner. Similarly to read restrictions, also the proposals aimed at enforcing write restrictions rely on ABE (e.g., [19]) or on selective encryption and key derivation (e.g., [20]). The first class of techniques are based on the combination of ABE with Attribute-Based Signature (ABS), while the second class of techniques rely on digital signatures and/or on HMAC functions.

VII. Conclusions

We have presented an approach enabling organizations to securely rely on cloud-based solutions for supporting corporate governance's internal controls and audit functions. By leveraging selective encryption and tags, our approach guarantees confidentiality and integrity of data subjects to audit as well as on controls on them, without requiring trust in the cloud provider. Only authorized parties will be able to access data and perform control functions, also being accountable for them. All parties involved in the ICA process, as well as the organization, will be able to assess integrity of the data and function execution, hence detecting possible misbehavior of the provider itself. Our work leaves space for extensions such as the management of conflict of interests in the enforcement of controls and the support for delegation.

Acknowledgments

The authors would like to thank Luca Saccagi for discussions on the problem. This work was supported in part by the EC under grant agreement 825333 (MOSAICrOWN) and the Italian MIUR under PRIN project HOPE.

References

[1] 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.

[2] 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.

[3] 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.

[4] 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 *The New Codebreakers: Essays Dedicated to David Kahn on the Occasion of His 85th Birthday*, P. Ryan, D. Naccache, and J.-J. Quisquater, Eds. Springer, 2016.

[5] 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.

[6] R. Popa, C. Redfield, N. Zeldovich, and H. Balakrishnan, "CryptDB: Protecting confidentiality with encrypted query processing," in *Proc. of SOSP*, Cascais, Portugal, Oct. 2011.

[7] S. De Capitani di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, and P. Samarati, "Integrity for join queries in the cloud," *IEEE TCC*, vol. 1, no. 2, pp. 187–200, Jul.-Dec. 2013.

[8] R. Merkle, "A certified digital signature," in *Proc. of CRYPTO*, Santa Barbara, CA, USA, Aug. 1989.

[9] B. Zhang, B. Dong, and H. Wang, "CorrectMR: Authentication of distributed SQL execution on MapReduce," *IEEE TKDE*, 2019, preprint.

[10] A. Sahai and B. Waters, "Fuzzy identity-based encryption," in *Proc. of EUROCRYPT*, Aarhus, Denmark, May 2005.

[11] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy Attribute-Based Encryption," in *Proc. of IEEE S&P*, Oakland, CA, May 2007.

[12] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," in *Proc. of CCS*, Alexandria, VA, Oct.-Nov. 2006.

[13] S. Yu, C. Wang, K. Ren, and W. Lou, "Achieving secure, scalable, and fine-grained data access control in cloud computing," in *Proc. of INFOCOM*, San Diego, USA, March 2010.

[14] Z. Wan, J. Liu, and R. H. Deng, "HASBE: A hierarchical attribute-based solution for flexible and scalable access control in cloud computing," *IEEE TIFS*, vol. 7, no. 2, pp. 743–754, Apr. 2012.

[15] 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.

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

[17] E. Bacis, S. De Capitani di Vimercati, S. Foresti, S. Paraboschi, M. Rosa, and P. Samarati, "Mix&Slice: Efficient access revocation in the cloud," in *Proc. of CCS*, Vienna, Austria, Oct. 2016.

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

[19] 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.

[20] 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.