2013 IEEE International Conference on Cloud Computing Technology and Science

Access control enforcement on query-aware encrypted cloud databases

Luca Ferretti, Michele Colajanni, and Mirco Marchetti
University of Modena and Reggio Emilia
Email: {luca.ferretti, michele.colajanni, mirco.marchetti}@unimore.it

***Abstract*—The diffusion of cloud database services requires a lot of efforts to improve confidentiality of data stored in external infrastructures. We propose a novel scheme that integrates data encryption with users access control mechanisms. It can be used to guarantee confidentiality of data with respect to a public cloud infrastructure, and to minimize the risks of internal data leakage even in the worst case of a legitimate user colluding with some cloud provider personnel. The correctness and feasibility of the proposal is demonstrated through formal models, while the integration in a cloud-based architecture is left to future work.**

I. Introduction

The cloud Database as a Service (DBaaS) [1] is a successful paradigm where the database engine and the storage devices are located in some cloud infrastructure. This scheme allows a cloud customer organization, called *tenant*, to outsource data storage and computation and to leverage availability, scalability and pay-per-use that typically characterize cloud services. Nevertheless, confidentiality concerns about data stored in remote locations and managed by unknown administrators are hindering a wide adoption of the cloud database paradigm [2], [3].

This paper addresses the issues related to *information leakage* due to the tenant personnel and *information theft* due to the provider personnel by combining encryption and access control mechanisms that represent the two pivotal schemes of any research data confidentiality solution. The interesting problem comes from the consideration that we are not referring to a basic cloud storage scenario where any robust encryption algorithm is applicable. Instead, we are considering relational databases where a legitimate tenant user should be able to issue SQL operations on data stored in a cloud infrastructure. This paper gives an original contribution to the state-of-the-art by proposing a new model for the combination of data encryption algorithms with encryption enforcement of database access control policies.

The access control mechanisms offered by relational databases allow a database administrator to specify which data can be accessed by each legitimate user according to some tenant security policies. The problem is that access control mechanisms are typically applied at the front-end of the database, hence any subject having direct access to the database storage could bypass them. A similar scenario occurs in the cloud database service where the cloud provider personnel has legitimate access to the whole back-end storage infrastructure hosting the databases. Hence, a malicious administrator could gain access to data outsourced by the customers of the cloud service without proper credentials [3].

Information confidentiality of data stored in cloud services can be achieved through encryption, but valid architectures must guarantee that all decryption keys are managed by the tenant and never by the cloud provider. For this reason, we cannot adopt encryption solutions that work for relational databases managed on private infrastructures (e.g., transparent data encryption [4]). Even the proposal of the same authors in [5] has some risks of information leakage because the encryption of the cloud database information is based on one master key shared by all legitimate users.

The original model proposed in this paper uses multiple keys and allows a database administrator to derive cryptographic keys starting from high level access control policies, and to distribute to each user only the decryption keys that are necessary to decipher all and only the information to which he has legitimate access. The enforcement of access control policies through encryption schemes guarantees that data outsourced to the public cloud database are always managed in an encrypted way, thus guaranteeing confidentiality for data in motion from and to the client-cloud, in use and at rest in the cloud. Moreover, it minimizes information leakage in the case of a user key loss or a compromised client machine, and even in the worst case scenario where a malicious but legitimate user colludes with a cloud provider personnel by disclosing his decryption keys. In such a case, a partial data leakage is inevitable but it is limited to the data set accessible by the malicious user, while a cloud provider adversary cannot gain additional information about other data that remain inviolable through standard attack techniques.

The remaining part of this paper is structured as following. Section II describes the main requirements and the constraints of the access control and encryption models. Section III contains the novel solutions, where feasibility and correctness of the encryption scheme are demonstrated through formal models. Section IV compares our proposal against existing solutions related to selective access control and database encryption techniques for the cloud. Section V outlines the main results and possible future work.

II. Encryption and access control models

We consider a typical scenario in which a tenant organization requires a database service from a public cloud DBaaS provider [1]. In the tenant, there is a *database administrator* role (DBA) and multiple *database users*. The DBA is a trusted subject. He has complete access on all database information, and is in charge of enforcing the *access control policies* of the tenant. Each tenant user has a different level of trust and a consequent authorization to access a specified subset of the

978-0-7695-5095-4/13 $31.00 © 2013 IEEE
DOI 10.1109/CloudCom.2013.132

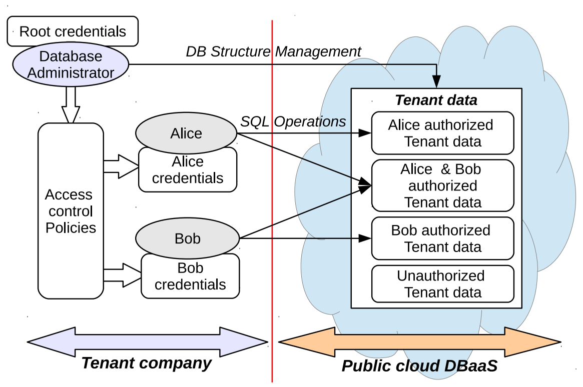

Fig. 1: Reference model for a multi-user encrypted cloud database.

database information. His database view is limited by the tenant access control policies that are implemented through authorization mechanisms. We also assume that the cloud provider is a *semi-honest* subject. This security model is also called *honest-but-curious* [6] and it is commonly adopted in literature (e.g., [5], [7]–[10]). In other words, we assume that the cloud employees execute correctly database protocols and mechanisms and do not modify stored data nor query results that would be noted by the tenant and would compromise the reputation of the cloud provider, but they could try to access tenant information stored in the cloud database.

Figure 1 evidences the considered scenario, where the cloud database stores all the *tenant data* while the tenant manages the following types of information: *access control policies*, *users credentials*, *root credentials*. *Tenant data* denote the information that is stored in the cloud database. They are accessed by *trusted database users*, such as Alice and Bob, through SQL operations. *Access control policies* are the rules of the tenant that define which data can be accessed by each user. For example, *Alice authorized (tenant) data* denote all and only tenant data on which Alice has legitimate access as defined by the access control policies. Users authorized data can be accessed by one or multiple tenant users, such as *Alice & Bob authorized data*. *Alice credentials* include all information that she requires to access and execute SQL operations on all and only her authorized data. The DBA is the only tenant subject that has access on *root credentials* granting complete access on cloud database information and data structures. A similar usage scenario poses several research challenges that we specify through the following requirements:

- $(R1)$: any confidential data outside the tenant domain must be encrypted;
- $(R2)$: users must be able to access only authorized data according to the access control policies. In such a way, even if user credentials are leaked, a malicious subject cannot get information that are inaccessible to that user;
- $(R3)$: encrypted tenant data stored in the cloud must allow the execution of SQL operations.

No existing proposals satisfy all previous three requirements. Here, we outline the three closest results, while an in-depth description is in Section IV. The SQL-aware encryption schemes presented in [7] satisfy $(R1)$ and $(R3)$, but do not enforce the standard database access control mechanisms on tenant data stored in the cloud as required by $(R2)$. The architecture proposed in [5] allows the execution of SQL operations from independent clients on an encrypted cloud database without intermediate servers, thus satisfying $(R1)$ and $(R3)$, because it uses one master key for all users. Encryption schemes described in [10] satisfy $(R1)$ and $(R2)$ because they enforce access control of users on encrypted data, but they consider a public-subscribe scenario in which SQL operations cannot be executed on encrypted data $(R3)$.

This paper proposes the first scheme enriched with encryption and access control schemes that satisfy all three requirements. Let us outline the main idea. All tenant data stored in the cloud database are encrypted through SQL-aware encryption schemes thus allowing the execution of SQL operations as required by $(R1)$ and $(R3)$. Tenant data are encrypted through different encryption keys, and each user is provided with unique credentials that allows him to obtain all and only the encryption keys associated with his authorized tenant data as required by $(R2)$. The basic concepts and notations related to access control and SQL-aware encryption are described in Sections II-A and II-B, respectively.

A. Access control

The objective of access control policies and mechanisms is simple: tenant users must access all and only authorized data on the cloud database, where authorizations are specified by the tenant access control policies. Standard access control mechanisms (e.g., [11]) that are implemented in all modern database management systems do not provide any confidentiality guarantee against honest-but-curious cloud providers. Indeed, they apply access control policies at the edge of the database infrastructure. Hence, data are not protected from malicious subjects that can bypass the edge controller by accessing the physical devices as in a cloud scenario. For this reason, we propose a novel encryption scheme that enforces standard relational database access control mechanisms by encrypting tenant data through multiple keys, that are distributed to the database users according to tenants access control policies (e.g., [10]). The formal notations of the access control model and of the encryption enforcement are presented below.

We represent the access control policies through the triple $(\mathcal{U}, \mathcal{O}, \mathcal{A})$, where $\mathcal{U}$ is the set of users, $\mathcal{O}$ is the set of objects, and $\mathcal{A}$ is the access matrix [11]. For each user $u \in \mathcal{U}$ and for each object $o \in \mathcal{O}$, there exists a binary authorization rule $a \in \mathcal{A}$ that defines if access to o by u is denied ($a_{u,o} = 0$) or allowed ($a_{u,o} = 1$). User u *capability list* cap_u is the set of objects to which u has authorized access. We remark that set of objects $\mathcal{O}$ must be modeled upon the structure of tenant data stored in the cloud database in order to allow definition of standard database access control mechanisms.

Let the set of resources $\mathcal{R}$ represent plaintext tenant data, $\mathcal{E}$ the set of encrypted tenant data, and $\mathcal{K}$ the set of decryption keys. We assume the existence of a decryption function $D : \mathcal{E} \times \mathcal{K} \longmapsto \mathcal{R}$ such that for each encrypted resource $e \in \mathcal{E}$, there exists a key $k \in \mathcal{K}$ that allows us to calculate

(a) Plaintext table

	c_1	–	c_n	–	c_N
R_1	$x_{1,1}$	–	$x_{1,n}$	–	$x_{1,N}$
R_j	$x_{j,1}$	–	$x_{j,n}$	–	$x_{j,N}$
R_J	$x_{J,1}$	–	$x_{J,n}$	–	$x_{J,N}$

	ψ_1	–	ψ_m^{n,ϕ_1}	ψ_{m+1}^{n,ϕ_2}	ψ_{m+2}^{n,ϕ_3}	–	ψ_M
R_1	$y_{1,1}$	–	$y_{1,m} = \phi_1(k^{n,1}, x_{1,n})$	$y_{1,m+1} = \phi_2(k^{n,2}, x_{1,n})$	$y_{1,m+2} = \phi_3(k^{n,3}, x_{1,n})$	–	$y_{J,M}$
R_j	$y_{j,1}$	–	$y_{j,m} = \phi_1(k^{n,1}, x_{j,n})$	$y_{j,m+1} = \phi_2(k^{n,2}, x_{j,n})$	$y_{j,m+2} = \phi_3(k^{n,3}, x_{j,n})$	–	$y_{J,M}$
R_J	$y_{J,1}$	–	$y_{J,m} = \phi_1(k^{n,1}, x_{J,n})$	$y_{J,m+1} = \phi_2(k^{n,2}, x_{J,n})$	$y_{J,m+2} = \phi_3(k^{n,3}, x_{J,n})$	–	$y_{J,M}$

(b) SQL-aware encrypted table

Fig. 2: Encrypting values of database tables through SQL-aware encryption algorithms.

$r = D(k, e)$, where $r \in \mathcal{R}$. For simplicity, we define $e_r \in \mathcal{E}$ and $k_r \in \mathcal{K}$ as the encrypted resource and the decryption key for resource $r \in \mathcal{R}$, i.e. $r = D(k_r, e_r)$. We define a *user u keyring $\mathcal{K}_u \subseteq \mathcal{K}$* as the set of all the decryption keys that user u owns, and *user accessed resources $\mathcal{R}_u$* as the set of all and only the resources that u is able to decrypt using the keys included in $\mathcal{K}_u$. The encryption scheme correctly enforces tenant access control policies if all users keyrings include the keys that decrypt all and only resources included in their capability lists [10], [11].

$$\forall u \in \mathcal{U}, \forall r \in \mathcal{R}, k_r \in \mathcal{K}_u \iff r \in cap_u \tag{1}$$

B. SQL-aware encryption scheme model

The requirement $(R3)$ imposes that the proposed architecture must be able to execute SQL operations on encrypted data. To this purpose, we refer to SQL-aware encryption schemes proposed in [5], [7], [12]. SQL-aware schemes make use of several encryption algorithms, each supporting only a subset of SQL operators. In realistic database workload (e.g., the standard TPC-C testbed), several different SQL operators are applied to the same column. Hence, a problem arises if no encryption algorithms exist that support all the SQL operators that have to be executed on a single column. The solution that is commonly adopted in literature [7], [13] is to create many encrypted versions of the same plaintext column. Each encrypted version is ciphered through a different SQL-aware encryption algorithm, such that each required SQL operator is supported by at least one of the encrypted versions. Here, we define formal notation of the schemes, and we remark constraints that we take into account for the design of the encryption enforcement scheme in Section III.

Let us consider the plaintext database table represented in Figure 2a, that is composed by N columns $c_1 \ldots c_n \ldots c_N$, J rows $R_1 \ldots R_j \ldots R_J$, and $N \times J$ plaintext values $x_{1,1} \ldots x_{j,n} \ldots x_{J,N}$. The corresponding encrypted table is represented in Figure 2b. It consists of M columns $\psi_1 \ldots \psi_m \ldots \psi_M$ $(M \geqslant N)$, the same number J of rows, and $M \times J$ encrypted values $y_1 \ldots y_{j,m} \ldots y_{J,M}$. Let $y = \phi(k, x), \phi \in \Phi$ denote encryption of the plaintext value x through the encryption key k, where ϕ is the SQL-aware encryption algorithm, and Φ is the set of all SQL-aware algorithms available in the system. An encrypted column ψ_m can also be represented as $\psi^{n,\phi}$ to denote that it includes values of the plaintext column c_n encrypted through the encryption algorithm ϕ. As an example, in Figure 2 the plaintext column c_n is associated to encrypted columns $\psi_m^{n,\phi_1}, \psi_{m+1}^{n,\phi_2}, \psi_{m+2}^{n,\phi_3}$ that make use of three different SQL-aware encryption algorithms $\phi_1, \phi_2, \phi_3 \in \Phi$, using encryption keys $k^{n,1}, k^{n,2}, k^{n,3}$, respectively. We remark that each encrypted column may have a different key size due to the different SQL-aware algorithms. Hence, each key is a stream of bits whose length depends on the SQL-aware algorithm.

We define Ψ as the set of all the encrypted columns. Authorizing access to an encrypted column $\psi^{n,\phi}$ implies accessing information included in all columns $\psi^{n,\phi^\star}, \forall \phi^\star \in \Phi_n$, where $\Phi_n := \{\phi \in \Phi : \psi^{n,\phi} \in \Psi\}$, because they store the same information of the plaintext column c_n. Hence, the proposed model imposes the finest authorization granularity to sets of encrypted columns that correspond to the same plaintext column. That is, if a user keyring includes the decryption key of a generic encrypted column, then he must also include the decryption keys of all the others encrypted columns associated to the same plaintext column.

$$\begin{aligned} &\forall u \in \mathcal{U}, \forall \psi^{n,\phi} \in \Psi, \\ &k^{n,\phi} \in \mathcal{K}_u \Rightarrow k^{n,\phi^\star} \in \mathcal{K}_u, \forall \phi^\star \in \Phi_n \end{aligned} \tag{2}$$

III. Scheme design

We present the novel scheme and related models in three phases that gradually enrich the previous step: Section III-A describes the model of the plaintext database structure; Section III-B defines the model of the encrypted database; Section III-C defines the scheme for the distribution of the users credentials.

A. Plaintext database model

We model the plaintext database through the following triple:

$$\mathbb{P} := (\mathcal{S}, \succ, \mathcal{R}) \tag{3}$$

where $(\mathcal{S}, \succ)$ is the partially ordered set (poset) of the database structures, and $\mathcal{R}$ is the set of resources that represents tenant data.

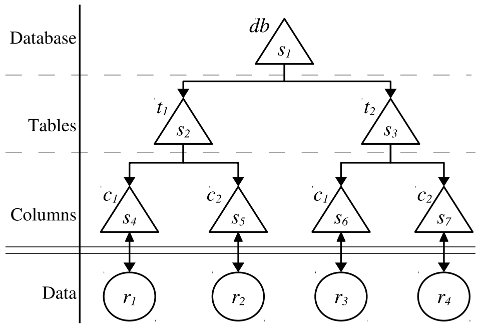

Fig. 3: The structure poset tree and the associated resources in the plaintext database model.

Each element $s \in \mathcal{S}$ is a structure of the database (e.g., a table, a column), and the ordering operator $x \succ y$, where $x, y \in \mathcal{S}$, denotes that x is an ancestor of y, and y is a descendant of x. If a third structure $z \in \mathcal{S} : x \succ z \succ y$ does not exist, then we use the notation $x \gtrdot y$, where x is a parent node of y, and y is a child node of x. We remark that a parent (child) is also an ancestor (descendant), while the opposite is not true. All inclusion relations between database structures are represented as parent-child relations in the poset (e.g., column c of the table t is represented by $t \gtrdot c$). Moreover, we define that each s is uniquely identified by a label ℓ_s.

Each element $r \in \mathcal{R}$ is the set of all information stored in a column of the database. If we model the structure poset as a hierarchical tree, then there is a 1:1 correspondence between each resource $r \in \mathcal{R}$ and each leaf of the poset tree. As an example, we refer to Figure 3 that represents the model of a plaintext database schema (s_1) containing two tables (s_2, s_3), each consisting of two columns (s_4, s_5, and s_6, s_7). Columns represent the leafs of the poset tree, and the set of data stored in each column are represented as a resource (i.e., r_1 represents the actual data stored in column s_4). The labels associated with structures are the actual names of the structures in the database, concatenated with the absolute path from the root of the structure poset. As an example, the label ℓ_{s_4} of structure s_4 is '$db.t_1.c_1$'.

We define that a structure $s \in \mathcal{S}$ associated to a resource $r \in \mathcal{R}$ is a parent of the resource r ($s \gtrdot r$). Moreover, all structures $s^\star \in \mathcal{S}$ that are ancestors of s ($s^\star \succ s$) are also ancestors of r ($s^\star \succ r$). We remark that a discretionary access control model can be defined on the model through the triple $(\mathcal{U}, \mathcal{S}, \mathcal{A})$ that extends the generic $(\mathcal{U}, \mathcal{O}, \mathcal{A})$ described in Section II-A. In other words, the objects $\mathcal{O}$ are implemented as plaintext database structures $\mathcal{S}$, and an authorization to a structure recursively apply to all the descendant structures. As an example, an authorization $a_{u,s} = 1$, where $u \in \mathcal{U}$ and $s \in \mathcal{S}$, authorizes u to access s and all descendant structures $s^\star \in \mathcal{S}, s \succ s^\star$ and descendant resources $r \in \mathcal{R}, s \succ r$.

B. Encrypted database model

We model the SQL-aware encrypted database through the set $\mathbb{E}$, that is an extension of $\mathbb{P}$ (3):

$$\mathbb{E} := (\mathcal{S}, \succ, \mathcal{R}, \mathcal{G}, \mathcal{V}, \Phi_R, \mathcal{K}, \mathcal{E}, \mathcal{T}, \theta, \gamma) \quad (4)$$

where:

$(\mathcal{S}, \succ, \mathcal{R})$ is the partially ordered set (poset) of the encrypted database structures, as already defined for the plaintext database model.

$\mathcal{G}$ is the set of the *access groups*, where each $g \in \mathcal{G}$ is a set of structures $\mathcal{S}_g \subset \mathcal{S}$. We assume also that each g is uniquely identified by a label ℓ_g.

$\mathcal{V}$ is the set of *derivation keys* [8]. Derivation keys are used to compute encryption keys, and each access group has exactly one derivation key. A user u that owns an authorization for the access group g is able to obtain the derivation key $v_g \in \mathcal{V}$ associated to g.

Φ_R is the set of the SQL-aware encryption algorithms used to encrypt resources $\mathcal{R}$. We assume that it is a subset of the algorithms available in the system $\Phi_R \subseteq \Phi$ (see Section II-B). We define that each algorithm $\phi \in \Phi_R$ is uniquely identified by a label ℓ_ϕ.

$\mathcal{K}$ is the set of encryption keys used to encrypt plaintext resources (see Section II-B).

$\mathcal{E}$ is the set of the *encryption groups*. Each encryption group $e \in \mathcal{E}$ is a set of resources $\mathcal{R}_e \subseteq \mathcal{R}$ that are encrypted through the same encryption key k_e and same SQL-aware encryption algorithm $\phi \in \Phi_R$.

$\mathcal{T}$ is the set of *tokens* [8], [14]. Each token $t \in \mathcal{T}$ is a public value that is used to compute derivation and encryption keys. Any tenant users can access all the tokens.

θ is a *derivation function* [15] that makes it possible to compute derivation keys, defined as:

$$\theta : \mathcal{V} \times \mathcal{G} \times \mathcal{T} \longmapsto \mathcal{V} \quad (5)$$

$$\forall (a, b) \in \mathcal{G} \times \mathcal{G} : a \gtrdot b \Rightarrow \exists!\, t \in \mathcal{T} : \theta(v_a, \ell_b, t) = v_b \quad (6)$$

γ is a derivation function that makes it possible to compute information associated with encryption groups, that are labels of the included structures and decryption keys. We define the function as:

$$\gamma : \mathcal{V} \times \mathcal{G} \times \Phi_R^n \longmapsto \mathcal{G}^n \times \mathcal{K}^n, n \geqslant 1 \quad (7)$$

$$\forall (a, B, \Phi_B), a \in \mathcal{G}, B \subseteq \mathcal{E}, \Phi_B \subseteq \Phi_R : \forall b \in B, a \gtrdot b$$
$$\Rightarrow \gamma(v_a, \ell_a, \ell_{\Phi_B}) = \{(\ell_b, K_b) : b \in B\} \quad (8)$$

As an example, we refer to the encrypted database shown in Figure 4. Encrypted database structures ($s_1 \ldots s_{10}$) are represented by triangles, access groups ($g_1 \ldots g_7$) by boxes with rounded corners, encrypted resources ($r_1 \ldots r_7$) by circles, and encryption groups ($e_1 \ldots e_6$) by boxes. In this example, there is one database schema (s_1) that contains two tables (s_2, s_3). The table s_2 contains four columns ($s_4 \ldots s_7$), and the table s_3 contains three columns ($s_8 \ldots s_{10}$). Each column is associated to the corresponding set of encrypted data (r_1 represents the actual data stored in column s_4). This scheme shows associations between access groups and structures, and between encryption groups and encrypted resources. As an example, access group $g_2 = \{s_2\}$, while $g_5 = \{s_5, s_6, s_7\}$. Similarly, we remark encryption groups $e_1 = \{r_1\}$ and $e_4 = \{r_4, r_5\}$.

Figure 5 refers to the same encrypted database represented by the Figure 4, but highlights the relations among access

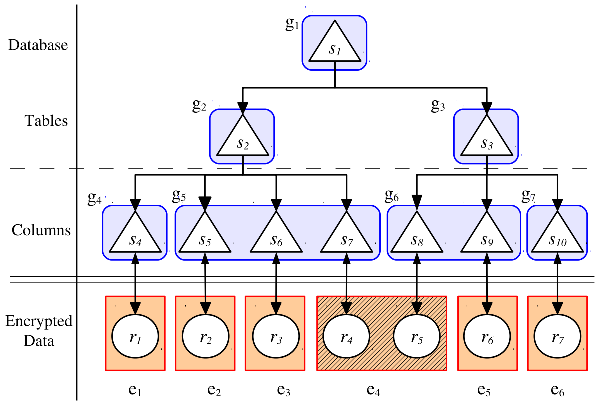

Fig. 4: Scheme of the structure of an encrypted database in the proposed model.

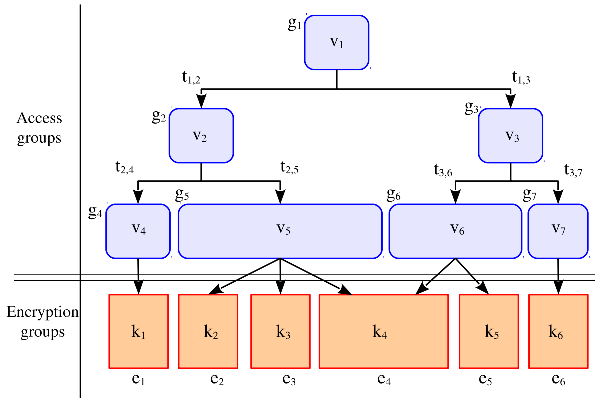

Fig. 5: Scheme of the access and encryption groups of an encrypted database in the proposed model.

and encryption groups. Here, each access group $g_1 \dots g_7$ is associated to a derivation key $v_1 \dots v_7$, respectively. Similarly, each encryption group $e_1 \dots e_6$ is associated to an encryption key $k_1 \dots k_6$. Each arrow represents a parent-child relationship between two access groups, or one access group and one encryption group. Each arrow that connects two access groups is associated to a token. As an example, a parent-child relationship $g_1 \succ g_2$ is associated to the token $t_{1,2}$.

For the sake of clarity, from now on we refer to the proposed models of plaintext (3) and encrypted databases (4) by using the following disambiguated notations.

$$\mathbb{P} := (\mathcal{S}_P, \succ, \mathcal{R}_P)$$
$$\mathbb{E} := (\mathcal{S}_E, \succ, \mathcal{R}_E, \mathcal{G}, \mathcal{V}_E, \Phi_E, \mathcal{K}, \mathcal{E}, \mathcal{T}_E, \theta, \gamma)$$

We define that for each plaintext structure $s_i \in \mathcal{S}_P$, there exists an associated access group $g_i \in \mathcal{G}$ in the encrypted database. In particular, we assume the existence of a cryptographic function h that allows to calculate an access group label ℓ_{g_i} from the access group derivation key ℓ_{v_i} and the plaintext structure s_i.

$$\forall s \in \mathcal{S}_P, \exists! g \in \mathcal{G} : \ell_g = h(v_g, \ell_s) \tag{9}$$

Thanks to this definition, we can conclude that discretionary access control policies $\mathcal{A}_P$ defined in the discretionary access control model $(\mathcal{U}, \mathcal{S}_P, \mathcal{A}_P)$ for a plaintext database $\mathbb{P}$ (see Section III-A) are enforced to the corresponding triple $(\mathcal{U}, \mathcal{G}, \mathcal{A}_E)$ defined for the encrypted database $\mathbb{E}$. That is, we can transparently transform any authorization rule $a_{u,s_i} \in \mathcal{A}_P$ defined on a plaintext plaintext structure s_i into a corresponding authorization rule $a_{u,g_i} \in \mathcal{A}_E$ defined on the corresponding access group g_i. The encrypted database enforces such access authorization, because a user u is authorized to access s_i if and only if he is able to calculate the derivation key associated with the corresponding access group v_{g_i}.

If a user u owns an authorization for the access group g, then u can access the access group derivation key v_g, and is implicitly authorized for all the access groups and the encryption groups descending from g. If a user u is implicitly authorized to the encryption group e, then he can decrypt all encrypted resources that are included in e. For example, a user u authorized for the access group g_2 owns an implicit authorization to g_4 and g_5. Hence, he is also implicitly authorized to access e_1, $e_2 \dots e_4$. Then, the user u can decrypt encrypted resources r_1, and $r_2 \dots r_5$. In this scenario, we show how u authorized on g_2 is able to decrypt r_3. User u already knows v_2, by definition of $\mathcal{V}$, $t_{2,5}$, because all tokens are public, and l_5, because he is implicitly authorized for g_5. Hence, he can compute v_5 through the formula (6): $v_5 = \theta(v_2, l_5, t_{2,5})$. After having computed v_5, u can use γ to compute the set of keys associated with encryption groups e_2, e_3, e_4 through the formula (8): $\{k_2, k_3, k_4\} = \gamma(v_5, \ell_5, \{\ell_{\phi_{e_2}}, \ell_{\phi_{e_3}}, \ell_{\phi_{e_4}}\})$. Information included in the encrypted resource r_3 can be decrypted through the key k_3.

C. Users credentials distribution scheme

We define the distribution scheme $\mathbb{D}$ that enforces the discretionary access control policies $(\mathcal{U}, \mathcal{G}, \mathcal{A})$:

$$\mathbb{D} := (\mathcal{U}, \mathcal{G}, \mathcal{A}, \mathcal{V}_{\mathcal{E}}, \mathcal{V}_{\mathcal{U}}, \mathcal{T}_U, \theta) \tag{10}$$

where:

$(\mathcal{U}, \mathcal{G}, \mathcal{A})$ represents the discretionary access control policy on the encrypted database, as described in Section III-B.

$\mathcal{V}_{\mathcal{E}}$ denotes the set of the encrypted database derivation keys, also included in the encrypted database model $\mathbb{E}$ (4).

$\mathcal{V}_{\mathcal{U}}$ is the set of the users derivation keys. Each element $v_u \in \mathcal{V}_{\mathcal{U}}$ is a secret derivation key owned by user $u \in \mathcal{U}$.

$\mathcal{T}_{\mathcal{U}}$ is the set of the users tokens. We denote $\mathcal{T}_u \subset \mathcal{T}_{\mathcal{U}}$ the set of tokens associated with user $u \in \mathcal{U}$. We remark that tokens can be accessed by all tenant users.

θ is a derivation function, as defined in (6).

The proposed scheme is based on the node-based key assignment schemes proposed in [8], [14], that we adapt to the considered cloud scenario. In our context, each tenant user $u \in \mathcal{U}$ owns a single derivation key v_u, and there exists a set of public tokens $\mathcal{T}_u \subset \mathcal{T}_{\mathcal{U}}$ associated with him. This user is able to calculate all and only derivation keys $v_g \in \mathcal{V}_u$ through the function θ (6), if ond only if there exists an associated token $t_{v_u,v_g} \in \mathcal{T}_u$.

This scheme assumes that the tenant DBA owns root credentials $\mathcal{C}_R$ that allow him to know any information stored in the encrypted database and all users credentials, because he requires them to calculate user tokens $\mathcal{T}_\mathcal{U}$ with respect to access control policies $\mathcal{A}$. Hence, we define that the set of the user tokens $\mathcal{T}_u$ must include a token t_{v_u,v_g} if and only if the plaintext database structure $s \in \mathcal{S}_P$ is included in the user capability list cap_u, and all the other structures included in the list are not ancestors of s.

$$\begin{aligned} &\forall u \in \mathcal{U}, \forall (s,g), s \in \mathcal{S}_P, g \in \mathcal{G}, \\ &s \in cap_u \wedge s^\star \nsucc s, \forall s^\star \in cap_u \iff t_{v_u,v_g} \in \mathcal{T}_u \end{aligned} \tag{11}$$

IV. Related Work

This paper addresses confidentiality issues that are the major problem affecting the widespread diffusion of cloud database services. The innovation of the proposed models and schemes is to enforce access control mechanisms on cloud databases while allowing the execution of SQL operations on encrypted data stored in the cloud that are accessible by any tenant cloud client. At the best of our knowledge, no existing proposal is able to satisfy both requirements. For example, there are encryption schemes that enforce access control mechanisms for cloud storage services [16], and other solutions that support concurrent accesses from independent clients [17]. Using query-aware encryption algorithms [7] allow a user to obtain all and only the requested data from the database, but that proposal is based on a trusted proxy that intercepts all operations between the tenant clients and the encrypted database, executes data re-encryption, and implements access control policies as in a privately managed infrastructure. No decryption keys are provided to users, but the trusted proxy affects service availability and elasticity of the cloud service. In [5] the same authors propose an architecture that avoids the necessity of a trusted proxy, thus allowing multiple clients to directly execute concurrent SQL operations on the encrypted database, but encryption is based on one master key. This solution simplifies implementation and administration of the architecture, but it would not guarantee the acceptable levels of confidentiality required to face possible insider malicious users. For these reasons, in this paper we look for a solution that guarantees the execution of SQL operations in an encrypted cloud database, and prevents users to have the same decryption key regardless their access privileges. Previous research in access control enforcement through encryption in [10] cannot be naively adopted because it refers to a publish-subscribe scenario, where the architecture requires to maintain lots of data on the user clients side. This scheme does not work in collaborative scenarios where multiple users can insert and update data, and concurrent SQL operations must be managed.

V. Conclusions

We propose a novel encryption scheme integrated with an access control mechanism that guarantees confidentiality of information stored in cloud databases. Unlike state-of-the-art proposals, the proposed scheme allows a customer company to encrypt all stored and transmitted data, to enforce standard database access control mechanisms where each tenant user has a different secret key, and to support the execution of SQL operations on encrypted data stored in a public cloud provider. This solution guarantees data confidentiality against a semi-honest cloud provider and limits the risk of information leakage due to internal users, even against the theft of access credentials, and the possibility that an internal user colludes with a cloud employee. This paper defines the overall idea and the formal models that demonstrate the correctness and feasibility of the proposed scheme. The integration of the proposal into cloud-based architectures is left to future work.

References

[1] H. Hacigümüş, B. Iyer, and S. Mehrotra, "Providing database as a service," in *Proc. of the 18th IEEE International Conference on Data Engineering*, February 2002.

[2] W. Jansen and T. Grance, "Guidelines on security and privacy in public cloud computing," Tech. Rep. NIST Special Publication 800-144, 2011.

[3] Cloud Security Alliance, "The notorious nine: Cloud computing top threats in 2013," https://cloudsecurityalliance.org/the-notorious-nine-cloud-computing-top-threats-in-2013, September 2013.

[4] Oracle, "Oracle transparent data encryption," http://www.oracle.com/technetwork/database/options/advanced-security/index-099011.html, September 2013.

[5] L. Ferretti, M. Colajanni, and M. Marchetti, "Distributed, concurrent, and independent access to encrypted cloud databases," *IEEE Transactions on Parallel and Distributed Systems*, 2013.

[6] O. Goldreich, *Foundations of Cryptography: Volume 2, Basic Applications*. Cambridge university press, 2004.

[7] R. A. Popa, C. M. S. Redfield, N. Zeldovich, and H. Balakrishnan, "CryptDB: protecting confidentiality with encrypted query processing," in *Proc. of the 23rd ACM Symposium on Operating Systems Principles*, October 2011.

[8] 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 *Proceedings of the 33rd international conference on Very large data bases*, September 2007.

[9] H. Hacigümüş, B. Iyer, C. Li, and S. Mehrotra, "Executing sql over encrypted data in the database-service-provider model," in *Proc. of the 2002 ACM SIGMOD international conference on Management of data*, June 2002.

[10] E. Damiani, S. De Capitani di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, and P. Samarati, "Key management for multi-user encrypted databases," in *Proc. of the ACM workshop on Storage security and survivability*, November 2005.

[11] P. Samarati and S. De Capitani di Vimercati, "Access control: Policies, models, and mechanisms," in *Foundations of Security Analysis and Design*. Springer, 2001.

[12] S. Tu, M. Kaashoek, S. Madden, and N. Zeldovich, "Processing analytical queries over encrypted data," in *Proc. of the 39th International Conference on Very Large Data Bases*, August 2013.

[13] L. Ferretti, F. Pierazzi, M. Colajannni, and M. Marchetti, "Security and confidentiality solutions for public cloud database services," in *Proc. of the 7th International Conference on Emerging Security Information, Systems and Technologies*, August 2013.

[14] J. Crampton, K. Martin, and P. Wild, "On key assignment for hierarchical access control," in *19th Computer Security Foundations Workshop*. IEEE, July 2006.

[15] M. J. Atallah, M. Blanton, N. Fazio, and K. B. Frikken, "Dynamic and efficient key management for access hierarchies," *ACM Transactions on Information and System Security (TISSEC)*, vol. 12, no. 3, 2009.

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

[17] A. J. Feldman, W. P. Zeller, M. J. Freedman, and E. W. Felten, "Sporc: group collaboration using untrusted cloud resources," in *Proc. of the 9th USENIX conference on Operating Systems Design and Implementation*, October 2010.