# Cloud Security: Issues and Concerns

**Authors**
*Pierangela Samarati*[*]
Università degli Studi di Milano, Italy
pierangela.samarati@unimi.it

*Sabrina De Capitani di Vimercati*
Università degli Studi di Milano, Italy
sabrina.decapitani@unimi.it

**Keywords**
cloud security
confidentiality
integrity
availability
secure data storage and processing

## Summary

The cloud has emerged as a successful computing paradigm allowing users and organizations to rely on external providers for storing and processing their data and making them available to others. An increasing important priority for the wide adoption and acceptance of cloud computing is the ability of data owners and users to have enforced and assess security guarantees. Guaranteeing security means ensuring confidentiality and integrity of data, accesses, and computations on them as well as ensuring availability of data and services to legitimate users and in compliance with agreements with the providers. In this chapter, we present an overview of the main security issues and concerns arising in the cloud scenario, in particular with respect to the storage, management, and processing of data.

## 1. Introduction

The rapid advancements in Information and Communication Technology (ICT) have enabled the emerging of the "cloud" as a successful paradigm for conveniently storing, accessing, processing, and sharing information. With its significant benefits of scalability and elasticity, the cloud paradigm has appealed companies as well as individuals, which are more and more resorting to the multitude of available providers for storing and processing data. Unfortunately, such a convenience comes at the price of loss of control of the owners of the data, and consequent security threats, which can limit the potential widespread adoption and acceptance of the cloud computing paradigm. On one hand, cloud providers can be assumed to employ basic security mechanisms for protecting data in storage, processing, and communication, devoting resources to ensure security that many individuals and companies may not be able to afford. On the other hand, data owners as well as users of the cloud lose control over data and their processing. ENISA lists loss of control and governance as a top risk of cloud computing (ENISA, 2009). The Cloud Security Alliance (CSA) lists data breaches and data loss as two of the top nine threats in cloud computing (CSA, 2013). Security threats can arise because of the new complexity of the cloud scenario (e.g., dynamic distribution, virtualization, and multi-tenancy), because data or computations might be sensitive and should be protected even from the providers eyes, or because providers might be not fully trustworthy and their - possibly lazy or malicious - behavior should be controlled.

The term cloud encompasses a variety of distributed computing environments, varying with respect to the architectural or trust assumptions and the services offered. In particular, the US National Institute of Standards and Technology (NIST) distinguishes four deployment models and three service models (NIST, 2011). The deployment models range from a *private* cloud, where the infrastructure and services are operated for a single organization and are maintained on a private network, to a *public* cloud, where the infrastructure is made available to the public and is owned by an organization offering cloud services. Ownerships and operation models between these two extremes are also possible, such as in a *community* cloud, where different companies with common objectives (e.g., business goals and security requirements) share the cloud infrastructure, and a *hybrid* cloud, composed of multiple clouds, which can be private, public, or community, under the control of one or more cloud providers, and with more stringent security requirements than a public cloud. Similarly, different service models, namely *IaaS* (Infrastructure as a Service), *PaaS* (Platform as a Service), and *SaaS* (Software as a Service), entail different responsibilities in enforcing security. The security and privacy issues to be addressed and the challenges involved can vary in different deployment and service models. For instance, a private cloud typically entails more control for the owner at the varying levels (applications, platform, and infrastructure) among data owners and providers.

In this chapter, we highlight security issues that need to be considered when using the cloud to offer or enjoy services, which are typically present, thought with possible variations, in the different models above. The chapter discusses security aspects that are more affected by the cloud paradigm, in particular in relationships to the *data security lifecycle*, reported in Figure 1 (CSA, 2011). Of course, complete protection requires also the use of others, perhaps more traditional, security techniques on which we do not further elaborate.

The chapter is organized in two main sections. Section 2 discusses how the classical confidentiality, integrity, and availability properties translate in the cloud. Section 3 presents an overview of the security issues and concerns to be addressed to ensure confidentiality, integrity, and availability in such a complex scenario. For each identified issue, we provide a description of the problem and challenges to be addressed together with possible existing solutions or directions.

## 2. Confidentiality, Integrity, and Availability in the Cloud

Security problems can be classified with the classical CIA (*confidentiality*, *integrity*, and *availability*) paradigm, which in the cloud can be interpreted as follows. Confidentiality requires guaranteeing proper protection to confidential or sensitive information stored or processed in the cloud. Depending on the requirements of the considered scenario, this can relate to any or all of: the data externally stored, the identity/properties of the users accessing the data, or the actions that users perform over the data. Integrity requires guaranteeing the authenticity of: the parties (users and providers) interacting in the cloud, the data stored at external providers, and of the response returned from queries and computations. Availability requires providing the ability to define and verify that providers satisfy requirements expressed in Service Level Agreements (SLAs) established between data owners/users and providers. The issues to be tackled, the challenges to be addressed, and the specific guarantees to be

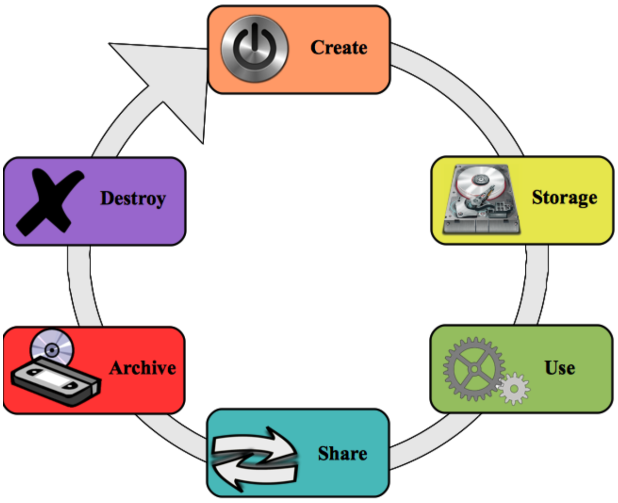


Figure 1: Data security lifecycle.

provided for ensuring satisfaction of the security properties above depend on the characteristics of the different scenarios. For instance, in a simple scenario, where an individual or a company uses the cloud simply for *archival/storage* purposes, problems to be addressed concern protecting confidentiality or integrity of data in storage and assessing satisfaction of Service Level Agreements, also ensuring correct enforcement of *create* and *destroy* operations. In a more complex scenario requiring execution of queries over data (*use*), the problem arises of executing queries as well as guaranteeing confidentiality and integrity of the dynamically computed results. The case where not only the owner (or a restricted set of trusted users) accesses the data (*share*) entails further complications such as the need to enforce access control restrictions over the data, ensure data integrity in presence of concurrent independent operations, and even ensure confidentiality of a user's actions with respect to other users. A further aspect that affects the issues to be addressed and possible applicable techniques are the trust assumptions – and consequent potential threats – on the providers involved in the storage and processing of the data, which could be *fully trusted*, *curious*, *lazy*, or *malicious*. Fully trusted providers can be assumed in cases of private clouds (or portions thereof) under complete and full control of the data owner. Curious providers refer to scenarios where the storage or processing involves sensitive information (data or actions on them) that should be maintained confidential to the providers themselves. Lazy providers refer to scenarios where the storing or processing providers might not be considered fully trustworthy for ensuring data or computation integrity or for providing the availability promised in the service level agreements. Finally, malicious (or byzantine) providers refer to cases where providers may intentionally behave improperly in the management, storage, and processing of the data, possibly compromising their confidentiality, integrity, or availability (this case accounts also for insider threats at the provider's side).

## 3. Issues and Challenges

The discussion in the previous section makes it clear that there is not a one-size-fits-all solution (not even a one-size-fits-all problem definition). There are instead different aspects, with related issues, challenges, and security controls that need to be considered and that can find application in different scenarios. In this section, we illustrate these issues and challenges, summarized in Figure 2.

| **Issue** | **Description** |
|---|---|
| *Protection of data at rest* | Guarantee confidentiality, integrity, and availability of data |
| *Fine-grained access* | Enable fine-grained retrieval and query execution on protected data |
| *Selective access* | Enable owner-regulated access control and authorization enforcement |
| *User privacy* | Support privacy of users accessing data and performing computations |
| *Query privacy* | Support privacy of users' actions in the cloud |
| *Query and computation integrity* | Enable assessment of correctness, completeness, and freshness of queries and computations |
| *Collaborative query execution with multiple providers* | Enable controlled data sharing for collaborative queries and computations involving multiple providers |
| *SLA and Auditing* | Specification and assessment of security requirements to be satisfied by providers |
| *Multi-tenancy and virtualization* | Provide confinement of different users data and activities in the shared cloud environment |

**Figure 2: Summary of cloud security issues.**

### 3.1 Protection of data at rest

A first basic problem that needs to be addressed when relying on the cloud for storing data is to guarantee protection (i.e., confidentiality, integrity, and availability) to the stored data themselves. With current solutions, users typically need to completely trust the cloud providers. In fact, although cloud providers apply security measures to the services they offer, such measures allow them to have full access to the data. For instance, Google Docs or Salesforce support encryption of the data both in transit and in storage but they also manage the encryption keys, and therefore users do not have direct control on who can access their data. Whenever data confidentiality needs to be guaranteed even to the provider's eyes, other solutions have to be considered. Solutions for protecting confidentiality in this, *honest-but-curious,* scenario typically require encrypting data before releasing them to the cloud providers (Figure 3(a)). For instance, services like Boxcryptor allow a user to encrypt her files locally before releasing them to a cloud provider such as Dropbox, Google Drive, and Microsoft SkyDrive. Encryption guarantees both confidentiality as well as integrity (as data tampering can be easily detected). For performance reasons, symmetric encryption is usually adopted. While encryption can be effective in many environments, it brings in several complications in scenarios where fine-grained retrieval of data needs to be supported (see Section 3.2). For this reason, recent approaches have put forward the idea of using *fragmentation*, instead of encryption, when what need to be maintained confidential are the associations among data values, in contrast to the values themselves (Ciriani et al., 2010). Fragmentation protects sensitive associations by splitting the concerned pieces of information and storing them in separate un-linkable fragments. Fragmentation can be applied in conjunction with encryption or by itself, resulting in different approaches (Figure 3(b-d)). In the *"two can keep a secret"* approach (Figure 3(b)), the data owner relies on two independent non-communicating providers, each of which stores a portion of the data, as much as possible in plaintext form, with encryption applied only to data values that either are sensitive by themselves or cannot be stored in the clear at any of the two providers without disclosing some sensitive associations. In the *"multiple un-linkable fragments"* approach (Figure 3(c)), only attributes with sensitive values are encrypted, while all other attributes are stored in the clear in as many fragments as needed, trying to avoid excessive fragmentation. In the *"keep a few"* approach (Figure 3(d)), nothing is encrypted and there is instead the involvement of a trusted party (typically the data owner) for storing and processing a limited amount of data that are sensitive by themselves or whose visibility would disclose some sensitive associations.

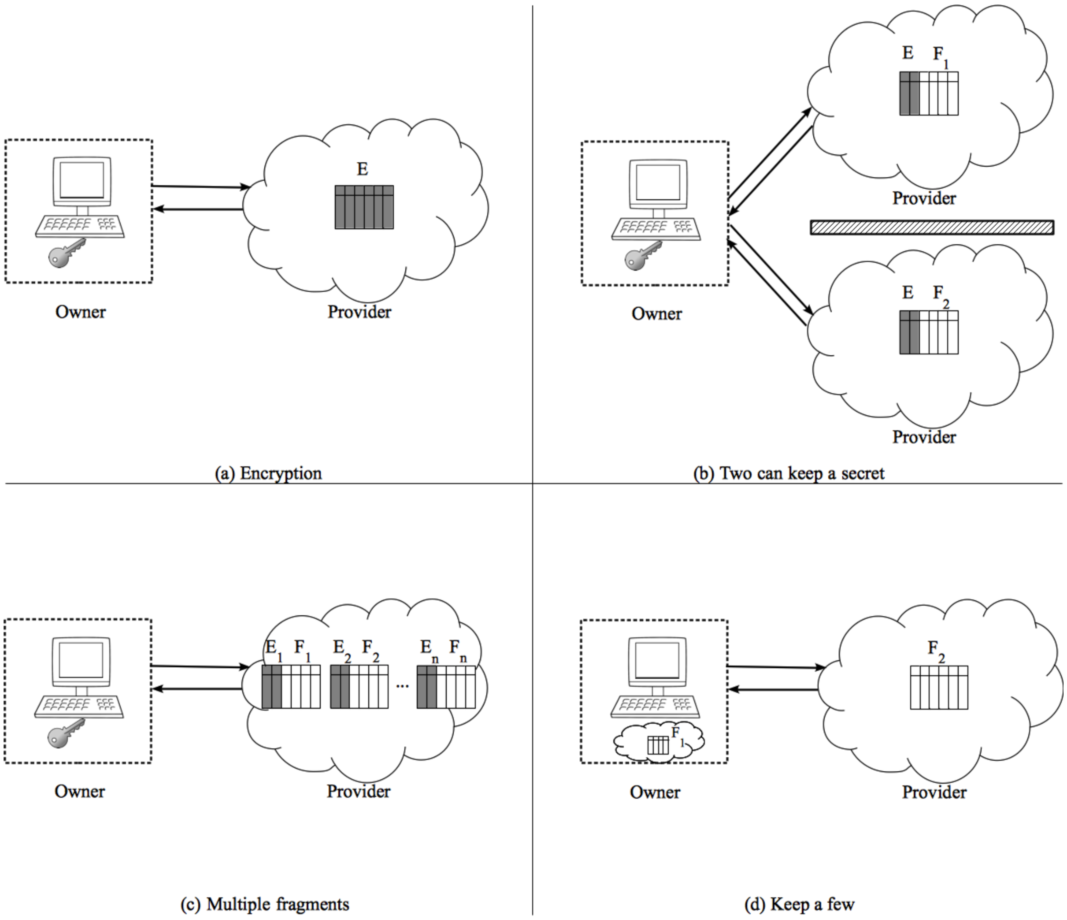


**Figure 3: Protection of data at rest with: encryption (a), fragmentation over two independent providers (b), fragmentation with un-linkable fragments (c), and fragmentation with the owner storing some of the data**

Ensuring integrity and availability of data in storage requires providing the data owners/users with the ability to verify that data have not been improperly modified or tampered with, and that their management at the provider side complies with the service level agreements. Integrity of data can be verified by employing signature schemes, where data are digitally signed so to make improper modifications on them detectable. Signatures provide a deterministic guarantee of data integrity. Probabilistic guarantees can be provided by the use of checks, such as *sentinels* used in Proof Of Retrievability (POR) solutions, which apply to encrypted data, or *homomorphic verifiable tags* used in Provable Data Possession (PDP) solutions, which apply to generic datasets. Availability of data in spite of failures or non compliance of providers can be guaranteed by employing classical replication techniques distributing data at different providers.

Protection of data entails also ensuring correct destruction of the data at the owner demand. The use of encryption under control of the owner can provide such a guarantee since possible remaining data copies would be intelligible without the proper key (Cachin et al., 2013).

### 3.2 Fine-grained access to data in the cloud

Maintaining confidentiality of the data even with respect to the providers storing or processing them implies, when data are protected with encryption, that the providers cannot decrypt the data for query execution. In applications where fine-grained access, typically query execution, needs to be supported, queries should then be evaluated on the encrypted data themselves. There are two lines of approaches for providing this ability. The first approach consists in performing *queries directly on the encrypted data,* where such a capability is made possible by specific cryptographic techniques (e.g., homomorphic encryption). The main drawbacks of these approaches, applicable typically for keyword searches or very basic operations, remain the limited kinds of accesses supported and the computational complexity of the execution, which make them not applicable in many real life scenarios. Other solutions enabling execution of SQL queries directly on encrypted data while guaranteeing more support for operations and efficiency rely on different layers of encryption, each supporting specific operations. An example is CryptDB (Popa et al., 2011), where each relation is encrypted at the column level with different onion layers of encryption, each supporting the execution of a specific SQL operation. Whenever the CryptDB proxy server receives an SQL query, it determines the onion layer needed for its execution. If the encrypted data are not already at the required onion layer, the proxy sends to the provider the key of the onion layer enabling the provider to strip off the other layers and execute the query. The second approach consists in attaching to the encrypted data some metadata (*indexes*) that are then used for fine-grained information retrieval and query execution. For instance, in a relational table where tuples are encrypted, different indexes can be specified for the different attributes on which conditions might need to be evaluated. Indexes should be well related to the data behind them, so to be precise and effective for query execution, but at the same time should not leak information on such data. Such a protection should be guaranteed from *static observations* (observation of the encrypted and indexed data in storage) as well as *dynamic observations* (observation of the queries in execution on such data). Different kinds of indexes have been investigated, including *direct indexing* (providing a one-to-one correspondence between plaintext and index values), *bucket*- or *hash-based indexing* (providing a many-to-one correspondence between plaintext and index values), and *flat indexing* (providing a one-to-many correspondence between plaintext and index values). Other types of indexes have been investigated in relation to tree-based data structures, and order-preserving or homomorphic encryption solutions, for providing support of range queries or aggregate functions. Different approaches to indexes provide different protection guarantees as well as different support for, and performance in, query execution. For instance, the many-to-one correspondence in bucket and hash-based approaches, where multiple plaintext values collide to the same index, and the flat indexing, where all different index values have the same number of occurrences, provide better protection of the confidentiality of the indexing with respect to direct indexing, at the price however of a more complex query process. Also, indexing approaches based on order preserving encryption provide support for range queries, but are exposed to some information leakage.

Query execution over encrypted and indexed data typically involves a trusted client application translating the plaintext query $Q$ in a query $Q_p$ to be sent to the provider and query $Q_c$ performing some post-processing for decrypting data and removing possible tuples originated by collisions in the index function and not belonging to the result (Figure 4).

### 3.3 Selective access to data in the cloud

In many scenarios access to data is selective, meaning different users (or groups thereof) should enjoy different views and accesses over the data. When data are stored

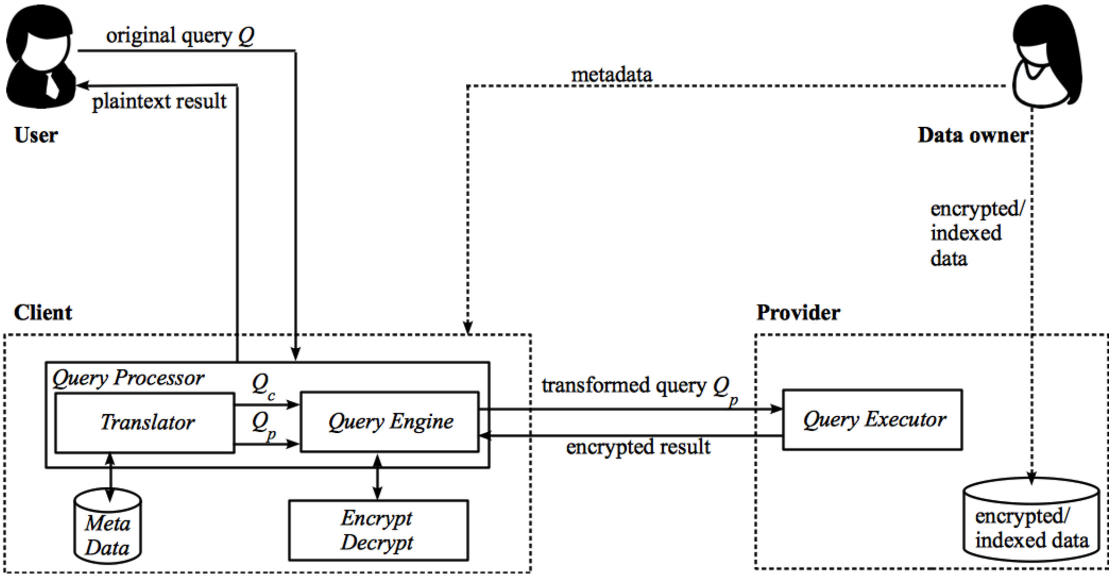
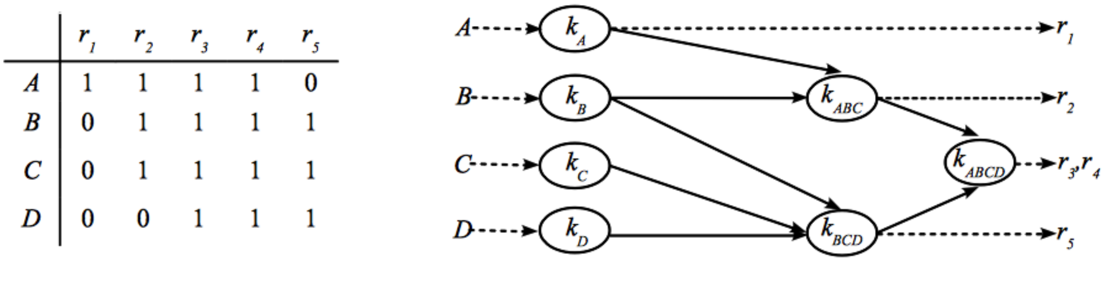


**Figure 4: Query evaluation over outsourced (encrypted/indexed) data: the user query Q is translated by a trusted client in a query $Q_p$ to be executed by the provider and a query $Q_c$ to be executed at the client side over $Q_p$'s result once decrypted.**

in the cloud, the problem arises of how to enforce such access control restrictions on them. For instance some cloud storage services (e.g., Amazon S3 and Google Cloud Storage) support the definition of access control lists for regulating access to data. The enforcement of such access control policy is however delegated to the cloud provider. In many scenarios this solution is not possible since the access control policy, just like the data, might be confidential and therefore should not be disclosed to the provider (note also that even authorizations to access data could leak information on the data themselves, therefore potentially compromising the protection enforced by encryption). Also, outsourcing access control to the cloud requires complete trust in the enforcing providers, as data protection would be completely in their hands (and providers could collude with users to acquire – and improperly grant – unauthorized access to data). On the other hand, having the data owner mediate every access request, to ensure only authorized accesses are granted, is clearly impractical and inapplicable. A promising approach to delegate access control to the cloud while not requiring complete trust in the providers relies on combining access control and encryption, that is, encrypt data with different keys, depending on the authorizations holding on them. Enforcing access control policies via encryption entails some challenges: users should not be required to hold many keys for the different resources they can access; at the same time every resource should be maintained only once (different replicas encrypted with different keys should be avoided as their management would clearly be impractical). This problem can be solved by employing *key derivation methods*, by which users can derive keys from a single key assigned to them and public tokens. Access control can then be enforced by properly organizing the keys in a hierarchy reflecting authorizations, or better the access control lists (ACLs) of resources, where the key corresponding to an ACL allows deriving – via one or more tokens – the keys associated with all ACLs that are superset of it. This way a user is able to derive, from her key and public tokens, all (and only) the keys that are needed to access resources that she is authorized to access (see Figure 5).

Updates to the access control policy can require changing the key with which resources have been encrypted, and therefore the need to download the resources from the cloud and release a newly encrypted version of them. Such a burden can be



|   | $r_1$ | $r_2$ | $r_3$ | $r_4$ | $r_5$ |
|---|---|---|---|---|---|
| *A* | 1 | 1 | 1 | 1 | 0 |
| *B* | 0 | 1 | 1 | 1 | 1 |
| *C* | 0 | 1 | 1 | 1 | 1 |
| *D* | 0 | 0 | 1 | 1 | 1 |

(a) access matrix

(b) key derivation hierarchy

**Figure 5: An example of access control policy (1 represents authorized accesses) with four users and five resources (a) and of key derivation hierarchy enforcing it: solid lines represent public tokens, dotted lines represent the keys associated with users and resources (b).**

avoided by assuming some collaboration from the external providers in enforcing policy changes, having the providers apply a further level of encryption, called *over-encryption* (De Capitani di Vimercati et al., 2010) in addition to – and on top of – the one applied by the owner. To access a resource *r* (see Figure 6), a user needs to pass both the encryption imposed by the provider (SEL, Surface Encryption Layer) and the encryption imposed by the owner (BEL, Base Encryption Layer).

Alternative solutions to enforce access control in the cloud use *attribute-based encryption* (ABE) techniques, possibly combined with other cryptographic techniques such as proxy and lazy re-encryption (Yu, Lou, and Ren, 2012). ABE is a public-key encryption that regulates access to data according to descriptive attributes associated with the data themselves and/or users, and to policies defined over these attributes. ABE can be implemented either as Ciphertext-Policy ABE (CP-ABE) or as Key-Policy ABE (KP-ABE), depending on how attributes and policies are associated with data and/or users.

### 3.4 User privacy

In a cloud scenario there might be need to grant access to data to users not registered in the system without requiring such users to declare their identity.

In these scenarios, access control authorizations and enforcement should be based on properties of users (in contrast to their identity), typically provided by means of attributes within digitally signed certificates. Access control solutions supporting this new paradigm are referred to as *attribute-based, credential-based, or certificate-based access control*, to stress the departure from identity to consider instead certified properties in the access decisions, or *privacy-enhanced access control*, to stress the privacy offered by departing from user authentication. Several proposals have investigated different issues to be addressed in this context, including: the language for expressing authorizations, the access control engine for evaluating users' requests, the possible dialog and negotiation to be supported between providers and users, the support for users' preferences with respect to properties to be released for acquiring services, and possible secondary use restrictions. As for languages, early proposals typically investigated the use of logic-based approaches, while later approaches aimed at balancing the trade-off between expressiveness of the language and simplicity of (and hence ability to maintain control on) the specifications. Different strategies for the dialog between users and providers have been investigated, including multi-step negotiations. Even in this case, later proposals aimed at balancing the need to exchange information to establish trust between users and providers, and the simplicity of the dialog to make it suitable for practical applications. As for user preferences, while earlier approaches assumed users to regulate release of their

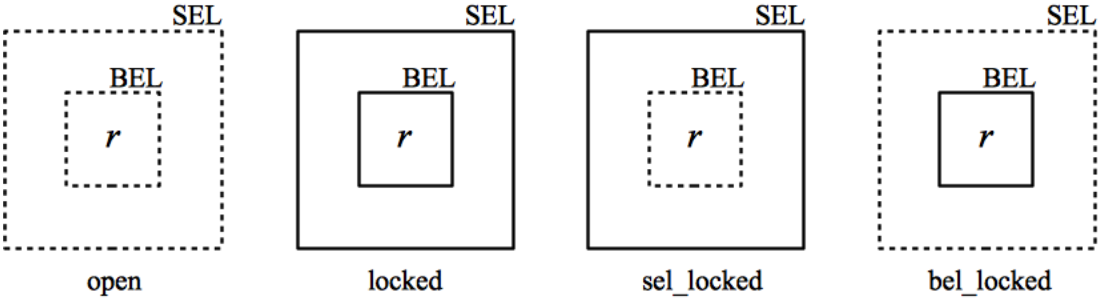


**Figure 6: Protection of resources with over-encryption. Every resource is encrypted first by the owner (BEL, Base Encryption Layer) then by the provider (SEL, Surface Encryption Layer). A resource is accessible (open) to a user only if she can pass both levels of encryption.**

credentials and properties with an access control approach similar to one adopted by the providers, more recent proposals have been investigating solutions specifically targeted to users and their natural way of thinking about preferences (Foresti and Samarati, 2012). Standards, such as XACML, have also being developed in these contexts supporting interoperation of access control policies.

### 3.5 Query privacy

In some scenarios what is confidential is not (or not only) data, or users' identities/properties, but also the accesses that users make on such data. In particular, confidentiality should be guaranteed, even from the provider's eyes, with respect to the fact that an access aims at a specific data (*access confidentiality*) or the fact that two accesses aim at the same data (*pattern confidentiality*). Traditional approaches for protecting access and pattern confidentiality are based on *Private Information Retrieval* (PIR) techniques that, assuming a database modeled as an N-bit string, provide protocols for users to retrieve the *i*-th bit in the string without disclosing to the server the specific bit accessed. In addition to the limitation of such a modeling and of the fact that they do not consider data confidentiality, PIR solutions suffer from high computational complexity and communication costs. Recent efforts, trying to make PIR more practical, have investigated the application of the Oblivious RAM, in particular with recent *practical ORAM* and *Path ORAM* solutions (Stefanov et al., 2013), and of a key-based hierarchical and dynamic data structure, called *Shuffle index* (De Capitani di Vimercati et al., 2011b). These proposals protect data confidentiality with encryption and protect access and pattern confidentiality by dynamically changing (*shuffling*), at every access, the physical location of the data, thus destroying the otherwise static correspondence between data and the physical blocks where they are stored. These approaches also employ a *cache* to maintain some data at the client side. Besides caching and dynamic allocation, Path ORAM assumes a tree-shaped data structure where nodes can contain, in addition to actual blocks, *dummy blocks* to guarantee that nodes have always the same size. The Shuffle index assumes that, at every access, additional fake searches, called *cover searches*, are executed together with the actual target search. Cover searches provide confusion to the provider with respect to the targeted block. At every access, the content of the blocks accessed (target/cover) and in cache is shuffled and rewritten. This changes dynamically the allocation of nodes, and the provider can only observe that some blocks have been read and written (Figure 7). By assuming a hierarchical value-based organization of the data (B+-tree with encrypted node content and with no pointer between leaves), the Shuffle Index is also able to support range-based queries.

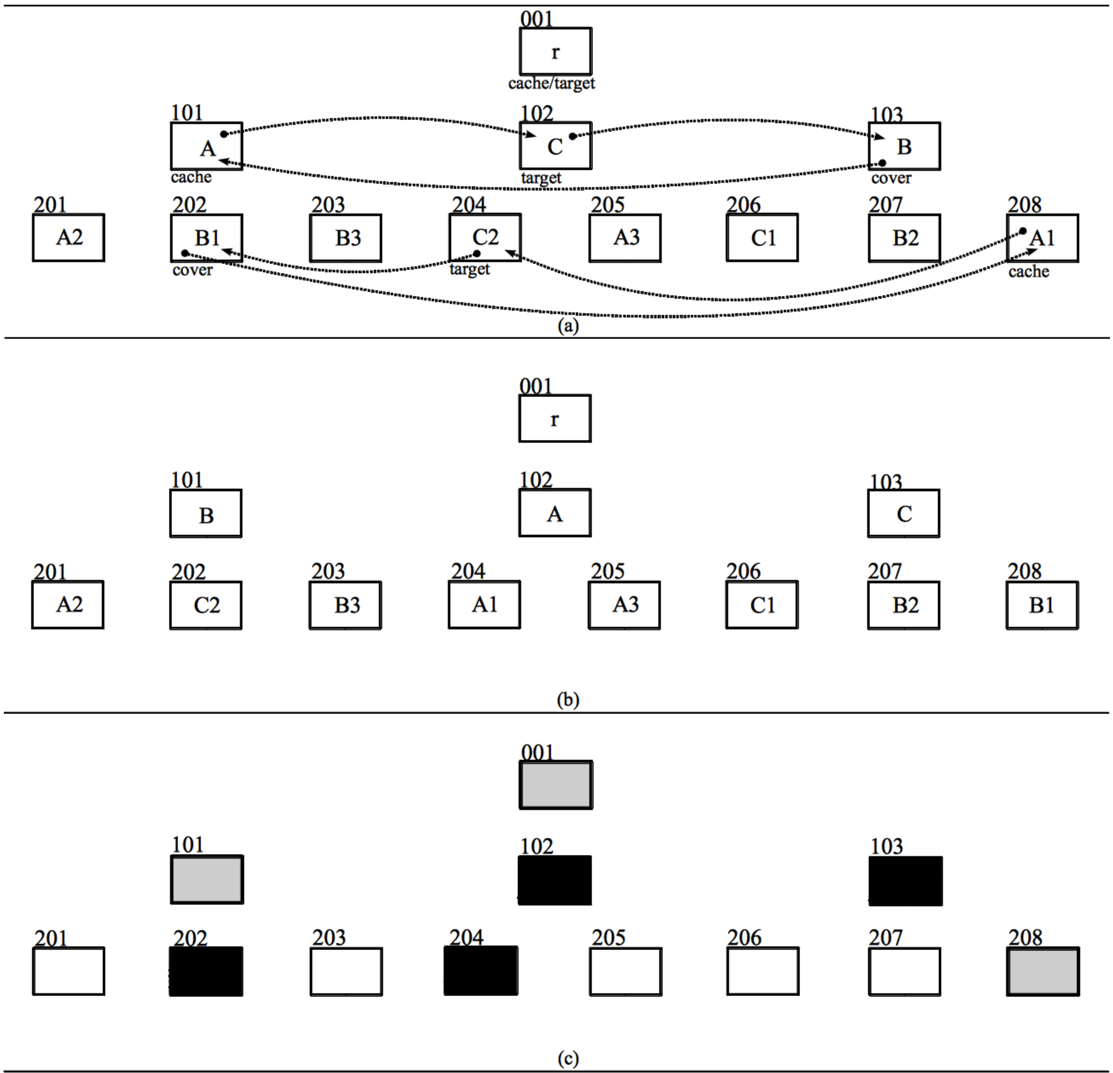


**Figure 7: Shuffle index: original structure with cache/target/cover and shuffling operations due to an access (a); resulting structure at the end of an access (b); server's view: blocks written (gray) or read and written (black) in the access execution (c).**

### 3.6 Query and computation integrity

In scenarios where queries/computations are performed by providers that are not fully trustworthy, the problem arises of providing data owners and/or users with the ability to assess that the result returned from a query/computation is *correct*, *complete*, and *fresh*. Correctness means that the result has been computed over the original data and the query/computation performed correctly. Completeness means that no data is missing from the result. Freshness means that the query/computation has been performed on the most recent version of the data. Most of the current approaches focus on providing guarantees of completeness and correctness, with some proposals complementing them with timestamps or periodical refreshing to provide freshness guarantees. Current solutions can be roughly classified in two categories: *deterministic* and *probabilistic*. Deterministic approaches are provided by authenticated data structures that, similarly to signature schemes for static data, permit to detect integrity violations with certainty. Examples of deterministic approaches for correctness/completeness are *signature chaining schemas* and *Merkle hash trees*. Signature chaining schemas allow the verification of the ordering among tuples and can then be used to verify the integrity of range queries where the selection condition is based on the attribute on which the signature schema has been applied. Merkle trees and their variations organize data within a tree-based structure over a given attribute (e.g., a search key). The result of a query with selection conditions on the attribute includes, in addition to the tuples belonging to the result, a verification object that allows the assessment of the integrity of the query (Figure 8). These authenticated data structures provide deterministic integrity guarantees but only for queries over the specific attribute/s on which the data structure has been organized. Techniques that have been applied, individually or in combination, for providing probabilistic guarantees include: insertion of *fake tuples* in the data, which if not retrieved in the query result signals an integrity compromise; *replication* of a portion of the data with replicas not recognizable as such, so that the presence of a duplicated data where the replica is missing signals an integrity compromise; and *pre-computation of tokens* associated with chosen query results, which allow the verification of the integrity of such queries. Probabilistic approaches, as their name says, provide only probabilistic guarantees: while the absence of an expected fake tuple or replica signals an integrity problem, its presence does not imply the integrity of the result since the providers might have just been lucky in not missing any of the checks inserted by the data owner. The probability of detecting an integrity compromise typically depends on the amount of controls (e.g., fake tuples, replicas inserted, or pre-computed tokens) enforced, where the more the control the higher the guarantees, but also the higher the performance overhead imposed for the verification. The involvement of multiple providers in the storage or computation complicates the scenario and requires devising additional controls. A possible solution to assess integrity of joins computed by an untrusted provider over data stored at two trusted storage servers assumes the cooperation of the storage servers to insert control information consisting of fake control tuples (*markers*) and duplicate tuples (*twins*) that if not present in the join result signals its incompleteness (De Capitani di Vimercati et al., 2014).

### 3.7 Collaborative query execution with multiple providers

Data stored and managed by different cloud providers may need to be selectively shared and accessed in a cooperative way. This scenario may see the presence of different providers as well as of different data owners. Exchange of data and collaborative computations should be controlled to ensure that information is not improperly accessed, released, or leaked. For instance, data stored at one provider might be released selectively only to specific providers and within specific contexts. Some solutions have addressed the specific problem of private and *secure multi-party computation*, which provide the ability of different parties to perform a collaborative computation learning only the query results and nothing on the inputs. Along the same line are solutions for computing *sovereign joins* over data, retrieving the result of a join operation over different tables, while guaranteeing confidentiality of the information not belonging to the join result. Recent approaches have also addressed a more general scenario where different parties (data owners or cloud providers) need to collaborate and share information for performing a *distributed query computation* with selective disclosure of data. Work has then investigated the problem of determining an efficient and safe execution plan for the query computation in which different providers collaborate releasing to others the information authorized and needed to compute the query result (De Capitani di Vimercati et al., 2011a).

### 3.8 Service Level Agreement and Auditing

A Service Level Agreement (SLA) is a contractual agreement that specifies the performance and availability guarantees that a cloud provider promises to deliver as

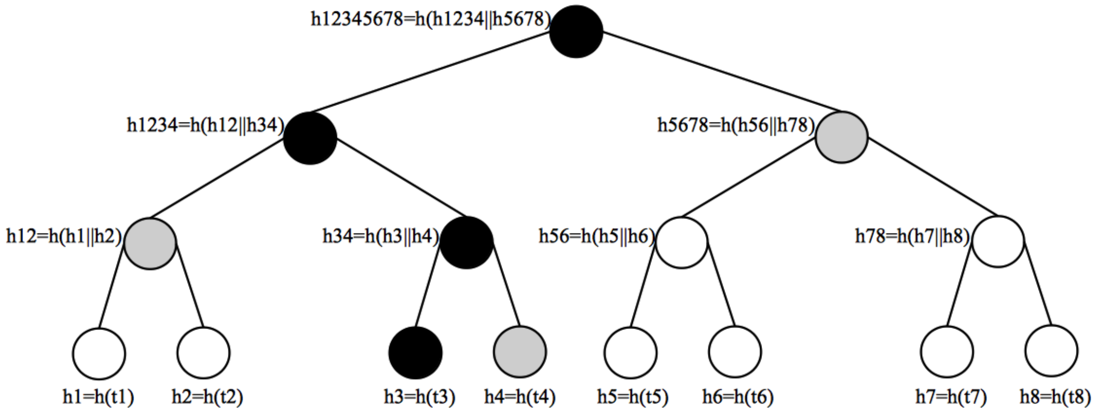


**Figure 8: A Merkle tree: every leave node is a hash of a tuple, internal nodes are hashed over the concatenation of their children. Colored node represent the integrity check assuming a query with result tuple t3: in gray the verification objects returned by the provider together with tuple t3, in black the hash computed for verification.**

well as penalties in the case of violations of the SLA. Due to the shared and dynamic nature of the cloud, cloud providers have to address several issues related to offering and managing SLAs, with different requirements coming from different users. Also, while in the past SLAs mainly focused on aspects related to the quality of the services offered (e.g., availability, response time, and fault resolution time), today they may also include the specification of the security guarantees, such as proofs on: the integrity of the stored data, their possession, their handling, or the application of specific security mechanisms (e.g., encryption or perimeter protection). In this contest, auditability of cloud providers, refers to the ability of users to verify full respect of the security guarantees declared in a SLA. Some proposals have presented solutions for verifying, for example, whether cloud providers are correctly storing data or correctly executing computation-intensive tasks on behalf of the users. In fact, lazy providers could delete some rarely accessed data or omit some computations to save resources. Some approaches apply Proof of Retrieval solutions as building blocks to allow users to verify that their data are: properly secured via encryption, intact, and retrievable. The correctness of the result of outsourced computations can be verified by applying the techniques for assessing integrity we have discussed previously.

### 3.9 Multi-tenancy and virtualization

Multi-tenancy refers to the ability to provide computing services to different users by using a common cloud infrastructure. Each user or company (i.e., a *tenant* of the cloud infrastructure) shares computation, memory, network, and storage resources, thus reducing the costs and improving the utilization of resources as well as the scalability and reliability. A basic mechanism enabling multi-tenancy in the cloud is *virtualization*, which creates a virtual version of, for example, an operating system, a storage device, or network resources, within a single physical system. Although virtualization brings great flexibility, it also introduces several security concerns that may have the *hypervisor* and/or the resident *virtual machines* as the main target. The hypervisor is a software component whose goal is to create and run the virtual machines. A compromised hypervisor can put at risk the confidentiality and integrity of the data managed by the virtual machines. Other security concerns can be related to the allocation and de-allocation of resources associated with virtual machines. In fact, improper leakages can result if the memory allocated to a virtual machine is not properly wiped before being reallocated to another virtual machine. Also, the communication, monitoring, modification, and migration of virtual machines can be a source of security concerns. In fact, due to the multi-tenant nature of cloud environments, there is the risk of improperly leaking information if the virtual resources allocated to different users are not perfectly isolated. Other aspects can be related to placement of virtual machine instances in the cloud, also supporting security constraints imposed by users, such as the request to not allocate given virtual machine instances to the same server (Jhawar, Piuri and Samarati, 2012).

## 4. Conclusions

With the rapid growth of cloud computing platforms and services, cloud security is becoming a key priority for all players (i.e., individuals, companies, and cloud providers). In this chapter, we presented an overview of security issues and concerns in cloud scenarios, illustrating their impact on the confidentiality, integrity, and availability properties and describing current solutions and possible challenges and directions.

## 5. References

Cachin, C and Haralambiev, K and Hsiao, HC and Sorniotti, A (2013). Policy-based Secure Deletion. *Proc. of the ACM Conference on Computer and Communications Security (CCS 2013),* Berlin, Germany.

Ciriani, V and De Capitani di Vimercati, S and Foresti, S and Jajodia, S and Paraboschi, S and Samarati, P (2010). Combining Fragmentation and Encryption to Protect Privacy in Data Storage. *ACM Transactions on Information and System Security (TISSEC).* 13(3):22:1-22:33.

Cloud Security Alliance – CSA, Top Threats Working Group (2013). The Notorious Nine - Cloud Computing Top Threats in 2013.
http://www.cloudsecurityalliance.org/topthreats.

Cloud Security Alliance – CSA (2011). Security Guidance for Critical Areas of Focus in Cloud Computing V3.0. http://www.cloudsecurityalliance.org/guidance/

De Capitani di Vimercati, S and Foresti, S and Jajodia, S and Paraboschi, S and Samarati, P (2011a). Authorization Enforcement in Distributed Query Evaluation. *Journal of Computer Security (JCS)*, 19(4):751-794.

De Capitani di Vimercati, S and Foresti, S and Paraboschi, S and Pelosi, G and Samarati, P (2011b). Efficient and Private Access to Outsourced Data. *Proc. of the 31st International Conference on Distributed Computing Systems (ICDCS)*, Minneapolis, Minnesota, USA.

De Capitani di Vimercati, S and Foresti, S and Jajodia, S and Paraboschi, S and Samarati, P (2014). Integrity for Join Queries in the Cloud. *IEEE Transactions on Cloud Computing (TCC)*, 1(2):187-200.

De Capitani di Vimercati, S and Foresti, S and Jajodia, S and Paraboschi, S and Samarati, P (2010). Encryption Policies for Regulating Access to Outsourced Data. *ACM Transactions on Database Systems (TODS)*, 35(2):12:1-12:46.

European Network and Information Security Agency – ENISA (2009). Cloud Computing: Benefits, Risks and Recommendations for Information Security. http://www.enisa.europa.eu/activities/risk-management/files/deliverables/cloud-computing-risk-assessment/at_download/fullReport

Foresti, S and Samarati, P (2012). Supporting User Privacy Preferences in Digital Interactions. *Computer and Information Security Handbook*, Vacca, JR (ed.), Morgan Kaufmann (2nd Edition).

Jhawar, R and Piuri, V and Samarati, P (2012). Supporting Security Requirements for Resource Management in Cloud Computing. *Proc. of the 15th IEEE International Conference on Computational Science and Engineering (CSE 2012)*, Paphos, Cyprus.

National Institute of Standards and Technology (2011). The NIST Definition of Cloud Computing. Special publication 800-145. http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf

Popa, RA and Redfield, CMS and Zeldovich, N and Balakrishnan, H (2011). CryptDB: Protecting Confidentiality with Encrypted Query Processing. *Proc. of the 23rd ACM Symposium on Operating Systems Principles (SOSP 2011)*, Cascais, Portugal.

Stefanov, E and van Dijk, M and Shi, E and Fletcher, C and Ren, L and Yu, X and Devadas, S (2013). Path ORAM: An Extremely Simple Oblivious RAM Protocol. *Proc. of the 20th ACM Conference on Computer and Communications Security (CCS 2013)*, Berlin, Germany.

Yu, S and Lou, W and Ren, K (2012). Data Security in Cloud Computing. *Handbook on Securing Cyber-Physical Critical Infrastructure*. Das, SK and Kant, K and Zhang, N (eds.), Morgan Kaufmann.