Supporting Users in Data Outsourcing and Protection in the Cloud

S. De Capitani di Vimercati(✉), S. Foresti, G. Livraga, and P. Samarati

Dipartimento di Informatica, Università degli Studi di Milano, 26013 Crema, Italy
{sabrina.decapitani,sara.foresti,giovanni.livraga,
pierangela.samarati}@unimi.it

Abstract. Moving data and applications to the cloud allows users and companies to enjoy considerable benefits. However, these benefits are also accompanied by a number of security issues that should be addressed. Among these, the need to ensure that possible requirements on security, costs, and quality of services are satisfied by the cloud providers, and the need to adopt techniques ensuring the proper protection of their data and applications. In this paper, we present different strategies and solutions that can be applied to address these issues.

1 Introduction

The 'Cloud' has emerged as a successful paradigm that enables users and companies to outsource data and applications to cloud providers, enjoying the availability of virtually unlimited storage and computation resources at competitive prices. An ever-growing number of cloud providers offer a variety of service options in terms of pricing, performance, and features. The advantages in adopting cloud services, however, come also with new security and privacy problems [24,54]. A first important problem consists in selecting, among the wide variety of cloud providers available on the market, the cloud provider most suitable for the needs of applications and data to be outsourced. This requires to properly model the requirements and/or preferences and to match such requirements with the characteristics and the service options offered by the cloud providers. Clearly, these requirements may differ for different users and/or for different data and applications that are moved to the cloud. Users therefore need flexible and expressive techniques supporting both the definition of their requirements and preferences, and the matching of these requirements with the characteristics of the different cloud providers (e.g., [6,28]).

Another problem (which often may lead users and companies to refrain from adopting the cloud for managing their data and applications) is related to ensuring proper protection of the outsourced data. As a matter of fact, when data and applications are moved to the cloud, they are no more under the direct control of their owner and must be properly protected from unauthorized parties or the cloud provider itself. Cloud providers may be *honest-but-curious* (i.e., trusted for managing data but not to access their content) or may even have a *lazy* or *malicious* behavior. Depending on the trust assumption that the user has on

M. Helfert et al. (Eds.): CLOSER 2016, CCIS 740, pp. 3–15, 2017.
DOI: 10.1007/978-3-319-62594-2_1

the provider running the selected cloud service, different problems might need to be addresses, including data confidentiality, integrity, and availability protection (e.g., [24]), the enforcement of access control (e.g., [16,19,34,59]), and the management of fault tolerance (e.g., [38–40]).

The remainder of this paper is organized as follows. Section 2 describes some approaches enabling users to express their requirements and preferences as well as possible strategies for selecting a cloud provider (or set thereof) that satisfies such requirements and preferences. Section 3 discusses the main data security issues that arise when using the cloud, and possible solutions to them. Finally, Sect. 4 concludes the paper.

2 Supporting Users in Cloud Provider Selection

Due to the growing number of cloud providers offering services that differ in their costs, security mechanisms, and Quality of Service (QoS), the research and industry communities have dedicated attention to the problem of improving the user experiences and the use of available cloud services. Industry security standards such as the Cloud Security Alliance Cloud Controls Matrix [13] have been proposed to allow cloud vendors and users to assess the overall security risks of cloud providers. Also several techniques have been recently proposed to assist data owners in selecting the cloud provider that better satisfies their needs. In this section, we provide an overview of such techniques.

User-side QoS. A natural strategy to compare a set of candidate cloud providers is based on measuring their Quality of Service (QoS) and comparing the results. Cloud providers typically publish indicators about the performance of their services in their Service Level Agreements (SLAs). Some of these indicators are used as parameters to measure the QoS of the provider (e.g., the response time). However, the value declared by the provider in its SLA (*provider-side QoS*) can differ from the value observed by a user (*user-side QoS*). Also, different users can experience different user-side QoS values for the same provider. For instance, the response time observed by a user can differ from the one experienced by another user operating in a different geographical area because they operate on networks with different latencies. Therefore, if a user selects a cloud provider on the basis of the provider-side QoS, she may end up with a choice not optimal for her. To overcome this problem, some techniques introduced the idea of selecting the cloud provider(s) based on the user-side QoS (e.g., [61]). A precise evaluation of user-side QoS values can however be a difficult task. In fact, in many situations, it requires to measure the parameters of interest (e.g., the response time) by actually invoking/using the services offered by the provider. This practice may cause communication overheads and possible economic charges that might not be always acceptable. To solve this issue, QoS parameters could be predicted by defining an automated QoS prediction framework that considers past service usage experiences of other "similar users". Measured or estimated QoS parameters are finally used to rank all the (functionally equivalent) providers among which the user can choose (e.g., [61]).

User Requirements. A user may have different requirements (i.e., conditions) that cloud providers should satisfy to meet the user's needs (e.g., at least four backup copies of the outsourced data should be maintained), or preferences on the values of the characteristics of the cloud service (e.g., a user may specify a value for the required availability and for the response time). User requirements might depend, for example, on the specific application scenario (e.g., data that need timely retrieval have to be stored at a provider with negligible downtime and fast response time) as well as by laws and regulations (e.g., sensitive data have to be stored at a provider applying appropriate security measures).

A line of research has investigated the definition of approaches to select the cloud provider (or set thereof) that better satisfies all the users requirements (i.e., optimizing the values of the attributes of interest for the user). The proposed solutions are typically based on the presence of a trusted middleware/interface playing the role of a *broker* [33] in the system architecture. The broker takes both the requirements of the user and the characteristics of the candidate cloud providers (possibly expressed in a machine-readable format [52]) as input, and tries to find the best match between the user requirements and the characteristics of the cloud providers. This matching problem can become more complex when a user defines multiple, may be even contrasting, requirements. In this case, it is necessary to quantify the satisfaction of each requirement by each provider, and to properly combine the measures associated with each provider. Multi-Criteria Decision Making (MCDM) techniques have been proposed as one of the basic approaches to address such a problem (e.g., [51]). Among the solutions relying on MCDM techniques, SMICloud [30,31] adopts a hierarchical decision-making technique, called Analytic Hierarchy Process, to compare and rank cloud providers on the basis of the satisfaction of user requirements. SMICloud models user requirements as key performance indicators (KPIs) that include, for example, the response time of a service, its sustainability (e.g., environmental impact), and its economic costs. MCDM, possibly coupled with machine learning, has also been proposed as a method for selecting the *instance type* (i.e., the configuration of compute, memory, and storage capabilities) that has the best trade-off between economic costs and performance and that satisfies the user requirements (e.g., [47,55]). For each of the resources to be employed (e.g., memory, CPU), these proposals select the provider (or set thereof) to be used for its provisioning as well as the amount of the resource to be obtained from each cloud provider so that the user requirements are satisfied.

Besides expressing requirements over QoS indicators and KPIs, a user might wish to formulate her requirements as generic conditions over a non-predefined set of attributes characterizing the service delivery and the cloud provider. For instance, to obey security regulations, a user may require that sensitive data be stored only in servers located in a given country, even if the physical location of a server is not an attribute explicitly represented in the provider SLAs. The satisfaction of user-based requirements might depend on the (joint) satisfaction of multiple conditions expressed over the attributes declared by cloud providers. Also, certain characteristics of a cloud provider may *depend* on other

characteristics. For instance, the response time of a system may depend on the incoming request rate (i.e., the number of incoming requests per second), meaning that it can be ensured only if an upper bound is enforced on the number of requests per time unit. When checking whether cloud providers satisfy the user requirements it is also important to consider such dependencies, accounting for the fact that different services might entail different dependencies (e.g., two services with different hardware/software configurations might accept different request rates to guarantee the same response time). Recent approaches have designed solutions for establishing an SLA between a user and a cloud provider based on generic user requirements and on the automatic evaluation of dependencies existing for the provider (e.g., [26]). For instance, a dependency can state that a response time of 5 ms can be guaranteed only if the request rate is lower than 1 per second. The solution in [26] takes as input a set of generic user requirements and a set of dependencies for a provider. By adopting off-the-shelves Constraint Satisfaction Problem (CSP) solvers, this technique determines a *valid* SLA (if it exists), denoted vSLA, that satisfies the conditions expressed by the user as well as further conditions possibly triggered by dependencies. With reference to the example above, if the requirement of the user includes a value for the response time, then the generated vSLA will also include a condition on the request rate that the candidate provider should also satisfy. Given a set of requirements and a set of dependencies, different valid SLAs might exist. The approach in [25] extends the work in [26] by allowing users to specify preferences over conditions that can be considered for selecting, among the valid SLAs, the one that the user prefers. Preferences are expressed over the values that can be assumed by the attributes involved in requirements and dependencies (e.g., the one between the response time and the request rate). Building on the CSP-based approach in [26], these preferences are used to automatically evaluate vSLAs, ranking higher those that better satisfy the preferences of the user.

Multi-application Requirements. Many of the approaches for selecting cloud provider(s) operate under the implicit assumption that a user is moving to the cloud a single application at a time (or a set of applications with the same requirements). Hence, the user requirements reflect the application needs. However, if a user wishes to outsource multiple applications to a single cloud provider, the selection process may be complicated by the fact that different applications might have different, even conflicting, requirements. Conflicting requirements then need to be reconciled to find a cloud provider that better suits the needs of all the applications. For instance, for an application operating with sensitive/personal information, a user will likely have a strong requirement on the security measures applied by the cloud provider (e.g., encryption algorithms, access control), while for an application performing data-intensive computations on non-sensitive information the same user will be more likely interested in performance (e.g., processing speed and network latency). An intuitive approach for considering the requirements of multiple applications in the selection of a cloud provider consists in first identifying the provider that would be preferred by each application singularly taken, and then in selecting the provider chosen

by the majority of the applications. While such an approach would certainly choose a single provider, it may still leave the requirements of some applications completely unsatisfied. To overcome this problem, alternative approaches based on MCDM techniques (e.g., [6]) aim at selecting cloud provider(s) in such a way that the chosen provider(s) balances the satisfaction of the application requirements, thus ensuring not to leave any application unsatisfied. Given the requirements characterizing the needs of each application and a set of cloud plans (among those available from cloud providers), the solution in [6] first adopts a MCDM technique to produce, for each application, a ranking of the cloud plans. Then, a consensus-based process is adopted to select the plan that is considered more acceptable by all applications. In particular, the consensus-voting algorithm takes as input the rank of each application and chooses the plan that better balances the preferences of all applications. Note that this approach considers all applications, and rankings produced by them, to be equally important when computing a solution to reach consensus. This implies that the proposed solution only considers the position of the plans in the ranking and does not take into consideration their relative distance. The cloud provider chosen by applying this technique may not be the first choice of any of the considered applications, but the technique guarantees that no application is left completely unsatisfied. An interesting alternative to be investigated can consider the application requirements by different applications as a single set of requirements to be globally optimized, considering not only the rankings but the distance among plans, possibly evaluating also preferences among applications.

Fuzzy User Requirements. To express precise requirements on the characteristics that should be satisfied by cloud providers, a user needs to have a technical understanding of both the service characteristics and the needs of the applications. However, the identification of 'good' value(s) for a given attribute might be challenging for some users, due to either a lack of technical skills, or the difficulty in identifying the precise needs for their applications. Also, the choice of a specific value or of a precise threshold for an attribute may be an overkill in many situations, imposing a too strong constraint. As an example, a requirement imposing a minimum service availability of 99.995% would exclude a service with a guaranteed availability of 99.990%, which – while less desirable – may still be considered acceptable by the user. To provide users with higher flexibility in the definition of their requirements, some works have considered fuzzy logic for the specification and consideration of requirements (e.g., [15,28]). Fuzzy logic can help users in defining their requirements in a flexible way whenever these requirements cannot be expressed through crisp values over attributes or are not easily definable. For instance, it is simpler for a non-skilled user to generically require 'high availability' rather than specifying precisely which values are good and which are not. In this example, 'high availability' maps to a set of values for the availability attribute, which fit the definition of 'high availability' according to the fuzzy membership function to be applied. In fact, the user and the cloud provider must agree on the meaning of the fuzzy values that a user can use in the definition of her requirements. Fuzzy logic can be used also to address other

issues in cloud scenarios, including the evaluation of cloud service performances (e.g., [49]) and the allocation by the provider of its resources to users applications (e.g., [4,15,29,50]). In this context, the allocation of resources to applications needs to take into account several aspects (e.g., the performance of applications, users costs, energy consumption, and security). Hence, finding an allocation that optimizes all these aspects is a difficult task that is usually addressed through MCDM techniques. Also, fuzzy logic can be useful for supporting flexible reasoning in resource allocation, especially in dynamic scenarios where applications are frequently activated/deactivated.

3 Protecting Data in the Cloud

Moving data and applications to the cloud implies a loss of control of the data owner over them and consequent concerns about their security. Guarantee data and applications security requires to address several problems such as the protection of the confidentiality and integrity of data and computations as well as data availability (e.g., [16,19,34,38–40,59]), the enforcement of access control (e.g., [16,19,34,59]), and query privacy (e.g., [22,23]). In this section, we focus on the solutions addressing the confidentiality, integrity, and availability of data.

3.1 Data Confidentiality

The protection of the confidentiality of (sensitive) data is the first problem that has to be considered when storing data at an external cloud provider. Sensitive data must be protected from untrusted/unauthorized parties, including the storing cloud provider, which can be considered *honest-but-curious* (i.e., trusted to properly operate over the data but not to see their content). Current solutions addressing this problem are based on the (possibly combined) adoption of *encryption* and *fragmentation*.

Encryption. Wrapping data with a layer of encryption before outsourcing represents a natural and effective solution to protect the confidentiality of outsourced data [53]. Indeed, only the data owner and authorized users, knowing the encryption key, can access the plaintext data. Encrypting data before outsourcing them guarantees that neither the cloud provider nor external third parties (possibly gaining access to the provider storage devices) can access the data content in the clear. However, encryption makes query execution difficult because data cannot be decrypted at the provider side. To address this problem, different solutions have been proposed, including encrypted database systems supporting SQL queries over encrypted data (e.g., [5,48]) and indexes for query execution (e.g., [10,37]). CryptDB [48] is an example of encrypted database system that supports queries on encrypted data. The idea is to encrypt the values of each attribute of a relational table with different layers of encryption, computed using different kinds of encryption (i.e., random, deterministic, order-preserving, homomorphic, join, order-preserving join, and word search), which depend on the queries to be executed. For instance, the values of an attribute can be first

encrypted using an order-preserving encryption schema, then a deterministic encryption schema, and then a random encryption. Proceeding from the outermost layer to the innermost layer, the adopted encryption scheme provides weaker security guarantees but supports more computations over the encrypted data. As an example, if the cloud provider has to execute a GROUP BY on such attribute, the random encryption layer is removed since such encryption does not allow to determine which values of the attribute are equal to each other. Instead, deterministic encryption supports grouping operations, and therefore it is not necessary to remove the deterministic encryption layer.

An index is a metadata associated with the encrypted data that can be used by the cloud provider to select the data to be returned in response to a query. Indexing techniques differ in the kind of queries supported (e.g., [2, 14,37,57]). In general, indexes can be classified in three main categories: *(1) direct indexes* map each plaintext value to a different index value and vice versa (e.g., [14]); *(2) bucket-based indexes* map each plaintext value to one index value but different plaintext values are mapped to the same index value, generating collisions (e.g., [37]); *(3) flattened indexes* map each plaintext value to different index values, each characterized by the same number of occurrences (flattening), and each index value represents one plaintext value only (e.g., [57]).

Besides indexing techniques classified as discussed, many other approaches have been proposed. For instance, indexes based on order preserving encryption support range conditions as well as grouping and ordering clauses (e.g., [2,57]), B+-tree indexes [14] support range queries, and indexes based on homomorphic encryption techniques (e.g., [32,36]) support the execution of aggregate functions.

While promising, encrypted database systems and indexes present open problems, such as the possible information leakage and the still limited support for query execution (e.g., [9,45]).

Fragmentation. Approaches based on encryption for protecting data confidentiality work under the assumption that all data need protection. However, in many scenarios, what is sensitive is the association among data values, rather than values singularly taken. For instance, with reference to a relational table, while the name of patients or the possible values of illness can be considered public, the association among them (i.e., the fact that a patient has a given illness) is clearly sensitive. Confidentiality can be guaranteed in this case by breaking the association storing the involved attributes in separate (unlinkable) *data fragments* (e.g., [1,10–12,17]). The application of fragmentation-based techniques requires first the identification of the sets of attributes whose joint visibility (i.e., association) is considered sensitive. A sensitive association can be modeled as a *confidentiality constraint* corresponding to sets of attributes that should not be publicly visible in the same fragment. For instance, with respect to the previous example, confidentiality constraint $\langle Name, Illness \rangle$ states that the values of attribute *Name* cannot be visible together with the values of attribute *Illness*. Fragmentation splits attributes in different data fragments in such a way that no fragment covers completely any of the confidentiality constraint.

Fragments need to be unlinkable for non-authorized users (including cloud providers) to avoid the reconstruction of the sensitive associations. Different approaches have been proposed to fragment data, as summarized in the following.

- *Two can keep a secret* [1]. The original table is split in two fragments to be stored at two non-communicating cloud providers. Sensitive attributes (i.e., singleton confidentiality constraints) are protected by encoding (e.g., encrypting) them. Sensitive associations are protected by splitting the involved attributes in the two fragments. If an attribute cannot be placed in any of the two fragments without violating a confidentiality constraint, it is encoded. Encoded attributes are stored in both fragments. Only authorized users can access both fragments as well as the encoded attributes, and reconstruct the original relation by joining the two fragments through a common key attribute stored in both fragments.
- *Multiple fragments* [10,12]. The original table can be split into an arbitrary number of disjoint fragments (i.e., fragments that do not have any common attribute). The idea is that sensitive attributes are stored in encrypted form while sensitive associations can always be protected by splitting the involved attributes in different fragments. Each fragment stores a set of attributes in plaintext and all the other attributes of the original table in encrypted form. Authorized users know the encryption key and can reconstruct the original table by accessing a single fragment (any would work) at the cloud provider. The use of multiple fragments guarantees that all the attributes in the original relation that are not considered sensitive by themselves can be represented in plaintext in some fragments.

  Unlinkability among fragments is ensured by the absence of common attributes in fragments. Such a protection may however be put at risk by *data dependencies* among attribute values since the value of some attributes may disclose information about the value of others attributes. For instance, knowing the treatment with which an individual is treated can reduce the uncertainty over her disease. If these attributes are stored in different fragments, they could be exploited for (loosely) joining fragments, thus possibly violating confidentiality constraints. Data dependencies have then to be considered in the fragmentation design [17].
- *Keep a few* [11]. The original table is split into two fragments, one of which is stored at the data owner side. Sensitive attributes are stored at the data owner side while sensitive associations are protected by storing, for each association, at least one attribute at the data owner side. This approach permits to completely depart from encryption. An identifier is maintained in both fragments to allow the data owner to correctly reconstruct the original table. Since one fragment is kept by the data owner, an access request may require her involvement.

3.2 Data Integrity and Availability

In addition to data confidentiality, data integrity and availability are also critical aspects. Data integrity means that the data owner needs guarantees on the

fact that cloud providers (and non-authorized users) do not improperly modify the data without being detected. Verifying data integrity consists not only in verifying whether the stored data have not been tampered with or removed (integrity of stored data) but also in verifying the integrity of query results. The integrity of a query result means that the result is *correct* (i.e., the result is computed on the original data and is correct), *complete* (i.e., the computation has been performed on the whole data collection and the result includes all data satisfying the computation), and *fresh* (i.e., the result is computed on the most recent version of the data). Existing solutions addressing these issues can provide *deterministic* or *probabilistic* guarantees.

- *Deterministic techniques* provide guarantees of data integrity with full confidence. Techniques for the integrity of stored data can be based on hashing and digital signatures as building blocks (e.g., [8,35,44]). These solutions require data owners to access their data in the cloud to check their integrity (which may imply high communication overhead). Deterministic guarantees on the integrity of computation results can be achieved by building *authenticated data structures* on the data (e.g., Merkle hash trees [43,46] and skip lists [3,27]). Every computation result is then complemented with a *verification object* VO, extracted by the cloud provider from the authenticated data structure. By checking the VO, the requesting user can easily and efficiently verify if the computation result is correct, complete, and fresh. While the adoption of authenticated data structures has the advantage of providing full confidence on the integrity of computation results, they are defined on a specific attribute and hence provide guarantees only for computations over it.
- *Probabilistic techniques* offer only probabilistic integrity guarantees, but are more flexible than deterministic approaches. Traditional solutions providing probabilistic integrity guarantees of stored data are Proof of Retrievability (POR) and/or Provable Data Possession (PDP) schemes [7,41]. These solutions either include sentinels in the encrypted outsourced data (POR) or pre-compute tokens over encrypted or plaintext data (PDP) to provide the owner with a probabilistic guarantee that the data have not been modified by non-authorized parties.

  Probabilistic guarantees on the integrity of computation results can be obtained by inserting fake tuples as sentinels/markers in the original dataset before outsourcing (e.g., [42,60]) or by duplicating (twinning) a portion of the original dataset (e.g., [20,58]). If sentinels/markers and duplicates/twins are not recognizable as such by the cloud provider, their absence in the computation result signals to the requesting user its incompleteness. Clearly, the higher the number of marker/twin tuples the higher the probabilistic guarantees obtained. It is interesting to note that these probabilistic strategies can be jointly used as their protection guarantees nicely complement each other [20,21] and can also be extended to work in a MapReduce scenario [18].

Data availability in the cloud can be interpreted as the ability of verifying whether the cloud provider satisfies users' requirements. Typically, the expected

behaviors of a cloud provider can be formalized using a Service Level Agreement (SLA) stipulated between a user and the cloud provider itself. An SLA can include confidentiality, integrity, and availability guarantees that the provider undertook to provide. Some proposals have then investigated the problem of how users can verify whether a cloud provider satisfies the security guarantees declared in an SLA (e.g., [56]). Also the PDP and POR techniques previously discussed for data integrity can also be used for verifying whether the cloud provider stores the data as declared.

4 Conclusions

In this paper, we have discussed the problems of enabling users to select cloud providers that best match their needs, and of empowering users with solutions to protect data outsourced to cloud providers. In particular, we have described techniques that allow users to express their security requirements, possibly defined for multiple applications and also using fuzzy logic, and to ensure confidentiality, integrity, and availability of outsourced data.

Acknowledgments. This work was supported in part by the EC within the FP7 under grant agreement 312797 (ABC4EU), and within the H2020 under grant agreement 644579 (ESCUDO-CLOUD).

References

1. Aggarwal, G., Bawa, M., Ganesan, P., Garcia-Molina, H., Kenthapadi, K., Motwani, R., Srivastava, U., Thomas, D., Xu, Y.: Two can keep a secret: a distributed architecture for secure database services. In: Proceedings of CIDR 2005, Asilomar, CA, USA, January 2005
2. Agrawal, R., Kierman, J., Srikant, R., Xu, Y.: Order preserving encryption for numeric data. In: Proceedings of ACM SIGMOD, Paris, France, June 2004
3. Anagnostopoulos, A., Goodrich, M.T., Tamassia, R.: Persistent authenticated dictionaries and their applications. In: Proceedings of ISC 2001, Malaga, Spain, October 2001
4. Anglano, C., Canonico, M., Guazzone, M.: FC2Q: exploiting fuzzy control in server consolidation for cloud applications with SLA constraints. Concurrency Comput. Pract. Experience **22**(6), 4491–4514 (2014)
5. Arasu, A., Blanas, S., Eguro, K., Kaushik, R., Kossmann, D., Ramamurthy, R., Venkatesan, R.: Orthogonal security with cipherbase. In: Proceedigs of CIDR 2013, Asilomar, CA, USA, January 2013
6. Arman, A., Foresti, S., Livraga, G., Samarati, P.: A consensus-based approach for selecting cloud plans. In: Proceedings of IEEE RTSI 2016, Bologna, Italy, September 2016
7. Ateniese, G., Burns, R., Curtmola, R., Herring, J., Kissner, L., Peterson, Z., Song, D.: Provable data possession at untrusted stores. In: Proceedings of ACM CCS 2007, Alexandria, VA, USA, October/November 2007
8. Boneh, D., Gentry, C., Lynn, B., Shacham, H.: Aggregate and verifiably encrypted signatures from bilinear maps. In: Proceedings of EUROCRYPT 2003, Warsaw, Poland, May 2003

9. Ceselli, A., Damiani, E., De Capitani di Vimercati, S., Jajodia, S., Paraboschi, S., Samarati, P.: Modeling and assessing inference exposure in encrypted databases. ACM TISSEC **8**(1), 119–152 (2005)
10. Ciriani, V., De Capitani di Vimercati, S., Foresti, S., Jajodia, S., Paraboschi, S., Samarati, P.: Combining fragmentation and encryption to protect privacy in data storage. ACM TISSEC **13**(3), 22:1–22:33 (2010)
11. Ciriani, V., De Capitani di Vimercati, S., Foresti, S., Jajodia, S., Paraboschi, S., Samarati, P.: Selective data outsourcing for enforcing privacy. JCS **19**(3), 531–566 (2011)
12. Ciriani, V., De Capitani di Vimercati, S., Foresti, S., Livraga, G., Samarati, P.: An OBDD approach to enforce confidentiality and visibility constraints in data publishing. JCS **20**(5), 463–508 (2012)
13. Cloud Security Alliance: Cloud Control Matrix v3.0.1. https://cloudsecurityalliance.org/research/ccm/
14. Damiani, E., Capitani, D., di Vimercati, S., Jajodia, S., Paraboschi, S., Samarati, P.: Balancing confidentiality and efficiency in untrusted relational DBMSs. In: Proceedings of CCS 2003, Washington, DC, USA, October 2003
15. Dastjerdi, A.V., Buyya, R.: Compatibility-aware cloud service composition under fuzzy preferences of users. IEEE TCC **2**(1), 1–13 (2014)
16. De Capitani di Vimercati, S., Foresti, S., Jajodia, S., Livraga, G., Paraboschi, S., Samarati, P.: Enforcing dynamic write privileges in data outsourcing. Comput. Secur. **39**, 47–63 (2013)
17. De Capitani di Vimercati, S., Foresti, S., Jajodia, S., Livraga, G., Paraboschi, S., Samarati, P.: Fragmentation in presence of data dependencies. IEEE TDSC **11**(6), 510–523 (2014)
18. De Capitani di Vimercati, S., Foresti, S., Jajodia, S., Livraga, G., Paraboschi, S., Samarati, P.: Integrity for distributed queries. In: Proceedings of IEEE CNS 2014, San Francisco, CA, USA, October 2014
19. De Capitani di Vimercati, S., Foresti, S., Jajodia, S., Paraboschi, S., Samarati, P.: Encryption policies for regulating access to outsourced data. ACM TODS **35**(2), 12:1–12:46 (2010)
20. De Capitani di Vimercati, S., Foresti, S., Jajodia, S., Paraboschi, S., Samarati, P.: Integrity for join queries in the cloud. IEEE TCC **1**(2), 187–200 (2013)
21. De Capitani di Vimercati, S., Foresti, S., Jajodia, S., Paraboschi, S., Samarati, P.: Efficient integrity checks for join queries in the cloud. JCS **24**(3), 347–378 (2016)
22. De Capitani di Vimercati, S., Foresti, S., Paraboschi, S., Pelosi, G., Samarati, P.: Efficient and private access to outsourced data. In: Proceedings of ICDCS 2011, Minneapolis, Minnesota, USA, June 2011
23. De Capitani di Vimercati, S., Foresti, S., Paraboschi, S., Pelosi, G., Samarati, P.: Shuffle index: efficient and private access to outsourced data. ACM TOS **11**(4), 1–55 (2015). Article 19
24. De Capitani di Vimercati, S., Foresti, S., Samarati, P.: Managing and accessing data in the cloud: Privacy risks and approaches. In: Proceedings of CRiSIS 2012, Cork, Ireland, October 2012
25. De Capitani di Vimercati, S., Livraga, G., Piuri, V.: Application requirements with preferences in cloud-based information processing. In: Proceedings of IEEE RTSI 2016, Bologna, Italy, September 2016
26. De Capitani di Vimercati, S., Livraga, G., Piuri, V., Samarati, P., Soares, G.: Supporting application requirements in cloud-based IoT information processing. In: Procedings of IoTBD 2016, Rome, Italy, April 2016

27. Di Battista, G., Palazzi, B.: Authenticated relational tables and authenticated skip lists. In: Proceedings of DBSec 2007, Redondo Beach, CA, USA, July 2007
28. Foresti, S., Piuri, V., Soares, G.: On the use of fuzzy logic in dependable cloud management. In: Proceedings of IEEE CNS 2015, Florence, Italy, September 2015
29. Frey, S., Claudia, L., Reich, C., Clarke, N.: Cloud QoS scaling by fuzzy logic. In: IEEE IC2E 2014, Boston, MA, USA, March 2014
30. Garg, S.K., Versteeg, S., Buyya, R.: SMICloud: A framework for comparing and ranking cloud services. In: Proc. of IEEE UCC 2011, Melbourne, Australia, December 2011
31. Garg, S.K., Versteeg, S., Buyya, R.: A framework for ranking of cloud computing services. Future Gener. Comput. Syst. **29**(4), 1012–1023 (2013)
32. Gentry, C.: Fully homomorphic encryption using ideal lattices. In: Proceedings of STOC 2009, Bethesda, MA, USA, May 2009
33. Goscinski, A., Brock, M.: Toward dynamic and attribute based publication, discovery and selection for cloud computing. Future Gener. Comput. Syst. **26**(7), 947–970 (2010)
34. Goyal, V., Pandey, O., Sahai, A., Waters, B.: Attribute-based encryption for fine-grained access control of encrypted data. In: Proceedings of ACM CCS 2006, Alexandria, VA, USA, October/November 2006
35. Hacigümüs, H., Iyer, B., Mehrotra, S.: Ensuring integrity of encrypted databases in database as a service model. In: Proceedings of DBSec 2003, Estes Park, CO, USA, August 2003
36. Hacigümüs, H., Iyer, B., Mehrotra, S.: Efficient execution of aggregation queries over encrypted relational database. In: Proceedings of DASFAA 2004, Jeju Island, Korea, March 2004
37. Hacigümüs, H., Iyer, B., Mehrotra, S., Li, C.: Executing SQL over encrypted data in the database-service-provider model. In: Proceedings of SIGMOD 2002, Madison, WI, USA, June 2002
38. Jhawar, R., Piuri, V.: Fault tolerance management in IaaS clouds. In: Proceedings of IEEE-AESS ESTEL 2012, Rome, Italy, October 2012
39. Jhawar, R., Piuri, V., Samarati, P.: Supporting security requirements for resource management in cloud computing. In: Proceedings of IEEE CSE 2012, Paphos, Cyprus, December 2012
40. Jhawar, R., Piuri, V., Santambrogio, M.: Fault tolerance management in cloud computing: a system-level perspective. IEEE Syst. J. **7**(2), 288–297 (2013)
41. Juels, A., Kaliski Jr., B.S.: PORs: Proofs of retrievability for large files. In: Proceedings of ACM CCS 2007, Alexandria, VA, USA, October/November 2007
42. Liu, R., Wang, H.: Integrity verification of outsourced XML databases. In: Proceedings of CSE 2009, Vancouver, Canada, August 2009
43. Merkle, R.: A certified digital signature. In: Proceedings of CRYPTO 1989, Santa Barbara, CA, USA, August 1989
44. Mykletun, E., Narasimha, M., Tsudik, G.: Authentication and integrity in outsourced databases. ACM TOS **2**(2), 107–138 (2006)
45. Naveed, M., Kamara, S., Wrigh, C.: Inference attacks on property-preserving encrypted databases. In: Proceedings of CCS 2015, Denver, CO, USA, October 2015
46. Pang, H., Jain, A., Ramamritham, K., Tan, K.: Verifying completeness of relational query results in data publishing. In: Proceedings of SIGMOD 2005, Baltimore, MA, USA, June 2005

47. Pawluk, P., Simmons, B., Smit, M., Litoiu, M., Mankovski, S.: Introducing STRATOS: A cloud broker service. In: Proceedings of IEEE CLOUD 2012, Honolulu, HI, USA, June 2012
48. Popa, R., Redfield, C., Zeldovich, N., Balakrishnan, H.: Cryptdb: Protecting confidentiality with encrypted query processing. In: Proceedings of SOSP, Cascais, Portugal (2011)
49. Qu, L., Wang, Y., Orgun, M.A.: Cloud service selection based on the aggregation of user feedback and quantitative performance assessment. In: Proceedings of IEEE SCC 2013, Santa Clara, CA, USA, June/July 2013
50. Rao, J., Wei, Y., Gong, J., Xu, C.Z.: DynaQoS: Model-free self-tuning fuzzy control of virtualized resources for QoS provisioning. In: Proceedings of IEEE IWQoS 2011, San Jose, CA, USA, June 2011
51. Rehman, Z., Hussain, O., Hussain, F.: IaaS cloud selection using MCDM methods. In: Proceedings of IEEE ICEBE 2012, Hangzhou, China, September 2012
52. Ruiz-Alvarez, A., Humphrey, M.: An automated approach to cloud storage service selection. In: Proceedings of ACM ScienceCloud 2011, San Jose, CA, USA, June 2011
53. Samarati, P., De Capitani di Vimercati, S.: Data protection in outsourcing scenarios: issues and directions. In: Proceedings of ASIACCS 2010, Beijing, China, April 2010
54. Samarati, P., De Capitani di Vimercati, S.: Cloud security: issues and concerns. In: Murugesan, S., Bojanova, I. (eds.) Encyclopedia on Cloud Computing. Wiley, Chichester (2016)
55. Samreen, F., Elkhatib, Y., Rowe, M., Blair, G.S.: Daleel: Simplifying cloud instance selection using machine learning. In: Proceedings of IEEE/IFIP NOMS 2016, Istanbul, Turkey, April 2016
56. van Dijk, M., Juels, A., Oprea, A., Rivest, R., Stefanov, E., Triandopoulos, N.: Hourglass schemes: How to prove that cloud files are encrypted. In: Proceedings of ACM CCS 2012, Raleich, NC, USA, October 2012
57. Wang, H., Lakshmanan, L.: Efficient secure query evaluation over encrypted XML databases. In: Proceedings of VLDB 2006, Seoul, Korea, September 2006
58. Wang, H., Yin, J., Perng, C., Yu, P.: Dual encryption for query integrity assurance. In: Proceedings of CIKM 2008, Napa Valley, CA, USA, October 2008
59. Waters, B.: Ciphertext-policy attribute-based encryption: an expressive, efficient, and provably secure realization. In: Proceedings of PKC 2011, Taormina, Italy, March 2011
60. Xie, M., Wang, H., Yin, J., Meng, X.: Integrity auditing of outsourced data. In: Proceedings of VLDB 2007, Vienna, Austria, September 2007
61. Zheng, Z., Wu, X., Zhang, Y., Lyu, M.R., Wang, J.: QoS ranking prediction for cloud services. IEEE TPDS **24**(6), 1213–1222 (2013)