Securus: From Confidentiality and Access Requirements to Data Outsourcing Solutions

Jens Köhler and Konrad Jünemann

Karlsruhe Institute of Technology (KIT), Steinbuch Centre for Computing (SCC), Karlsruhe, Germany,
{jens.koehler,konrad.juenemann}@kit.edu

Abstract. To preserve data confidentiality in database outsourcing scenarios, various techniques have been proposed that preserve a certain degree of confidentiality while still allowing to efficiently execute certain queries. Typically, several of those techniques have to be combined to achieve a certain degree of confidentiality. However, finding an appropriate combination is not a trivial task, as expert knowledge is required and interdependencies between the techniques exist. Securus, an approach we previously proposed, addresses this problem. Securus allows users to model their requirements regarding the information in the outsourced dataset that has to be protected. Furthermore, queries that have to be efficiently executable on the outsourced data can be specified. Based on these requirements, Securus uses Integer Linear Programming (ILP) to find a suitable combination of confidentiality enhancing techniques and generates a software adapter. This software adapter transparently applies the techniques to fulfill the specified requirements and can be used to seamlessly outsource and query the data. In this paper, we present an outline of Securus and extend our previous work by highlighting the differences to other approaches in the field. Furthermore, we show how Securus can be extended to allow for more efficient solutions if the attacker's capabilities can be modeled by the user.

1 Introduction

Preserving the privacy of individuals in today's service landscape is an ongoing research topic that gains even more importance with the trend of service outsourcing. Besides the protection of personal information that is necessary to make access control decisions [3], enforcing the confidentiality of personal data that is processed by third parties constitutes a challenge [2,12]. Ensuring confidentiality when outsourcing databases is both necessary to protect sensitive information and to adhere to privacy laws in many cases. One approach to tackle this problem is to establish a trust relationship with the external provider using *Service Level Agreements* (SLAs) [13] or by relying on laws for being able to hold the external provider accountable [16]. However, in many cases either no trust relationship can be established or (regional) laws forbid relying on trust alone. In these cases, technical means have to be used to preserve confidentiality [14].

M. Hansen et al. (Eds.): Privacy and Identity 2014, IFIP AICT 421, pp. 139–149, 2014.

A naive solution that technically preserves data confidentiality is to completely encrypt the whole database prior to outsourcing. However, queries on entirely encrypted data cannot be efficiently executed. It is thus better to selectively apply encryption techniques and partitioning the database on *multiple*, non-colluding external providers, as this way a tradeoff between data confidentiality and efficient query execution can be achieved [4,7,10,11,15]. However, it requires expert knowledge to choose a suitable set of existing techniques for a given scenario that protect confidential information while maximizing query efficiency. Furthermore, the choice highly depends on which parts of the data are considered sensitive and how the data will be queried.

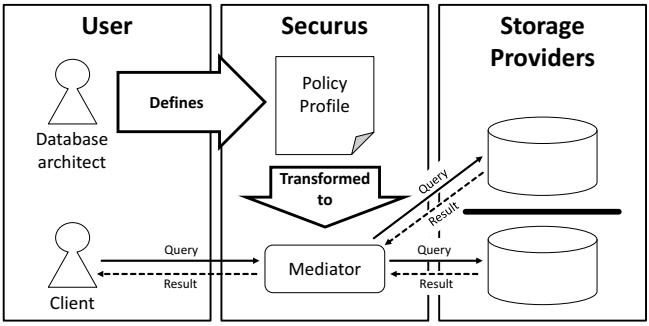

Fig. 1. Overview of the general Securus concept [14]

In this paper, we describe the *Securus* (**S**ecure and **E**fficient **C**loud **U**tilization **R**elying **U**pon **S**chemes) framework [14] that enables users to define which information of the outsourced data requires protection and which queries are executed on the data in a *policy profile* (see Figure 1). Furthermore, a set of available *storage providers* (SPs) to which the data will be outsourced can be specified.

Attacker model: The worst case Securus protects against is that each SP constitutes an honest-but-curious attacker or is compromised by one. We assume that the attackers a) do not collude and b) do not have background knowledge on data inserts/deletions/updates or incoming queries that can be used to infer information on data records. For instance, knowing that `SELECT...WHERE name='john'` is the most frequently executed query can be used to reveal encrypted names of the result records of the most frequently executed queries.

Based on the policy profile, Securus chooses a matching set of confidentiality enhancing techniques and generates a *mediator* software component that automatically applies them before outsourcing the data. Users can utilize mediators to transparently execute queries on the outsourced data, i.e., without taking the applied confidentiality enhancing techniques into account. Securus provides strong performance guarantees in terms of induced overhead during query executions, regardless of the outsourced dataset's composition.

The paper is structured as follows: We summarize related work and existing approaches Securus builds on in Section 2. In Section 3, we introduce the components of policy profiles and show how we adapted the building blocks of

Table 1. Catalog of security mechanisms for different queries and substitution categories. The ✓ symbol indicates that no security mechanisms need to be applied, × indicates that no security mechanisms are known that have the required properties.

Query	Plaintext	Deterministic Substitute	Probabilistic Substitute
Equality Selection	✓	Det. Encryption & Hash indices [7,4]	×
Range Selection	✓	Bucket Hash Indices [10]	×
Aggregation	✓	Homomorphic Encryption [11,15]	Homomorphic Encryption [11,15]

the policy profile (and thus the policy transformation process) to reflect the assumed attacker's knowledge more precisely compared to the already published version of Securus [14]. In Section 4 we show how to transform policy profiles into an outsourcing solution. As addition to our previous work [14] we highlight the differences between Securus and other existing approaches and provide an extended evaluation of our approach in Section 5. Finally, the paper is concluded in Section 6.

2 Related Work

In the *Database-as-a-Service* community, several contributions that share the goal of confidential data outsourcing exist. These can be categorized into approaches that encrypt data in a way that allows for the execution of specific queries [4,7,10,11,15], approaches that only encrypt parts of the data [17] and approaches that protect sensitive information by fragmenting the data on several non colluding providers [5,1].

Various *security mechanisms* have been proposed that encrypt attribute values while still allowing the efficient execution of queries containing *equality selections* (e.g. `...WHERE name='Doe'`) [4,7], *range selections* (e.g. `...WHERE age<30`) [10] and *aggregations* (e.g. `SELECT SUM(salary)...`) [11,15]. These security mechanisms can be categorized as shown in Table 1. We distinguish between mechanisms that map equal plaintext values on the same encrypted values (*deterministic substitute*) and mechanisms that map equal plaintext values on different encrypted values that cannot be distinguished (*probabilistic substitute*). While an attacker with background knowledge like the frequency distribution of an attribute can infer information from deterministic substitutes [4,8], no information can be inferred from probabilistic substitutes. However, in our model it is also impossible to evaluate equality or range selections on them. While cryptographic methods to search probabilistically encrypted data exist [18,6], the proposed methods require to "touch" every data record instead of using indexing structures or reveal information upon processing queries that can be used to distinguish records in a similar way deterministic ciphertexts do. Homomorphic

encryption schemes enable the storage provider to aggregate encrypted values before returning the query result to the client.

Additionally to these security mechanisms, the approach to protect sensitive attribute combinations by fragmenting the critical attributes across multiple SPs has been proposed [1]. For instance, while the attribute `name` and the attribute `salary` of a dataset may be viewed in plaintext by the attacker, the combination of them might be considered sensitive. Therefore, plaintext attribute values of `name` have to be stored by another SP than plaintext attribute values of `salary`.

CryptDB [17] is an approach that initially encrypts all attributes of the database that is outsourced to an external SP. Once a query needs to be executed, the key for the relevant attributes is passed to the SP. Thus, the SP incrementally unveils attributes and is able to execute queries efficiently. While providing performance guarantees regarding query execution, the protection of sensitive attributes cannot be guaranteed and depends on the query workload.

Securus makes use of both security mechanisms *and* attribute fragmentation to fulfill the user's requirements. Previous approaches [5,9] also propose to let the user specify which information contained in the data is sensitive and the query workload that will be executed on the data. They provide heuristics to find the best-effort attribute fragmentation that protects confidential relationships between attributes and produces as little overhead as possible for the given workload. Securus goes beyond true fragmentation and considers storing particular attributes at multiple SPs to allow for a more efficient query execution. Furthermore, Securus allows the user to specify hard performance requirements that must not be violated (cf. Section 5.1).

3 Policy Profiles

Before Securus can generate a mediator, the user has to specify his particular requirements in a *policy profile*. Besides the attributes that are contained in the dataset that is outsourced, the user defines the following three types of policies:

- **Access Policies** (APs) describe the queries that have to be efficiently executable. For instance, the AP `[Name,Salary]` expresses, that queries like `SELECT * FROM db WHERE Name=''xy'' AND Salary=10000` have to be efficiently executable. Besides equality selections, Securus also supports range selections and aggregation. For the sake of simplicity and space constraints we do not introduce them in this paper.
- A **Confidentiality Constraint** (CC) [1,5,9,19] constitutes a set of attributes that are considered sensitive if they are combined. For instance, the CC `[Name,Salary]` expresses that no name should be mapable on a salary value and vice versa. However, revealing the names *or* the salaries to an SP is tolerable. A special case are CCs containing a single attribute: `[Salary]` expresses that the salaries are sensitive and must not be revealed to any SP.
- **Inference Constraints** (ICs) specify a set of attributes. By including an attribute in the ICs it can be expressed that the assumed attacker is not

able to infer any information from deterministically encrypted values of the attribute[1]. In particular, this assumption holds for attributes that are guaranteed to be unique [8]. We are aware of the fact that ICs constitute very crude assumptions that users cannot confidently make in many scenarios. However, they provide an "interface" that enables us to explore a more fine-grained modeling of the attacker's capabilities in our future research. For instance, ICs could be derived from a more fine-grained attacker model specified by users.

4 Policy Transformation

4.1 General Concept

The workflow of an example mediator that was automatically generated by Securus is shown in Figure 2. To outsource a dataset (e.g. *Employees*), the attribute values of each record are encrypted probabilistically and put in the *main table* that can be stored by an arbitrary SP. As the probabilistic ciphertexts are indistinguishable for the SPs, no CC is violated. However, while it is possible to request specific records, no other queries can be executed based on the main table. Therefore, for each AP an *index table* that allows for efficient query execution is stored by at least one SP. Index tables contain attribute values in plaintext or protected by a security mechanism that does not prohibit the efficient execution of queries that comply to the AP (cf. Section 2). Thus, queries can be executed efficiently based on the according index table.

However, index tables might violate CCs. For instance, consider two index tables that store the attributes {`address`} and {`disease`} in plaintext, respectively. If those two index tables are stored by the same SP, this would violate the CC `[address,disease]`. The violation of CCs might be prevented by distributing the index tables on multiple, non colluding SPs and/or selectively applying security mechanisms on the attributes of the index table. For instance, in Figure 2, the CC `[name,salary]` is not violated as neither SP1 nor SP2 store both `name` and `salary` in plaintext or as deterministic ciphertexts that an attacker with – for instance – knowledge about the frequency of attribute values might infer plaintext values from. The CC `[name,age]` can be satisfied even though both attributes have to appear together in one index table to satisfy the AP `[name,age]`. This is due to the defined IC {`age`} that specifies, that it is assumed that the attacker cannot infer information from deterministically encrypted attribute values. Therefore, it suffices to encrypt `age` in index table 1 deterministically to comply with CC `[name,age]`.

4.2 Solving the Puzzle

Finding an appropriate distribution of the index tables on the SPs that minimizes the required application of security mechanisms and satisfies the defined

[1] For instance, if an attacker would know the most frequently occurring attribute value, he could reveal the most frequently occurring deterministic ciphertext.

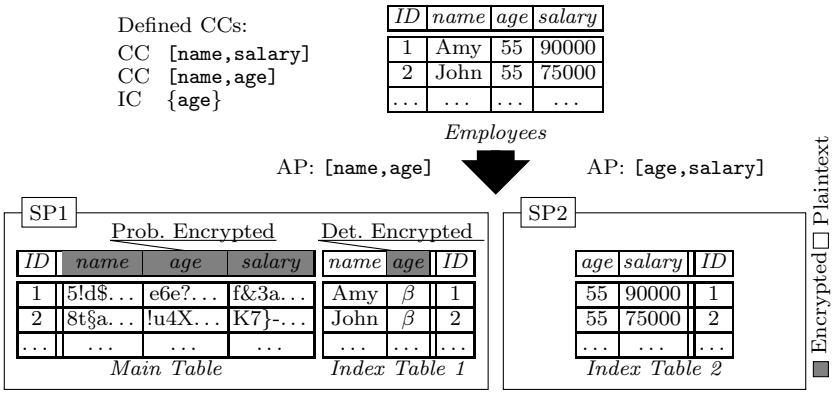

Fig. 2. Exemplary mediator generation

APs, CCs and ICs is not a trivial task and can be shown to be NP-hard. Securus reduces the problem of finding a suitable solution on an *Integer Linear Programming* (ILP) problem. ILP is a well understood mathematical model to specify optimization problems. ILP problems consist of constraints that define feasible solutions and an optimization criterion that characterizes the optimal solution.

While the exact formulation of the ILP problem is out of scope of this paper, we introduce the basic ideas. We utilize the constraints of the ILP to model the policies defined in the policy profile:

- **Access Policies**: To satisfy an AP, every attribute of the AP has to be stored in plaintext or as deterministic substitute by at least one SP to allow for an efficient execution of equality and range selections (cf. Table 1). The SP that stores all attributes of an AP may also store the AP's index table.
- **Confidentiality Constraints**: A CC is satisfied if at each SP at least one contained attribute is not revealable. To guarantee that, for each SP it must hold that at least one of the attributes contained in the CC is not stored at all, stored as probabilistic substitute or stored deterministically encrypted and contained in the specified ICs.

In order to apply security mechanisms only when necessary, we define the optimization criterion of the ILP as follows:

$$min \sum_{j \in SPs} \sum_{i \in Attributes} (d_{i,j} + p_{i,j}) \tag{1}$$

where

$$d_{i,j} = \begin{cases} 1, \text{if SP j stores deterministic substitutes of attribute i} \\ 0, \text{else} \end{cases}$$

$$p_{i,j} = \begin{cases} 1, \text{if SP j stores probabilistic substitutes of attribute i} \\ 0, \text{else} \end{cases}$$

The versatile nature of the ILP optimization criterion allows us to consider fine-grained performance differences of security mechanisms and database systems of the SPs in future work.

Once the ILP problem is solved by existing ILP solvers the solution of the ILP problem can be used to derive an optimal distribution of the index tables across the SPs and to determine which security mechanisms need to be applied on the attributes of the index tables. For instance, from Table 1 it can be deduced that if SP1 may store attribute *age* as deterministic substitute, hash indices can be used in an index table that should enable efficient queries that contain an equality selection on *age* (cf. Figure 2).

5 Evaluation

5.1 Benefit of Redundantly Storing Attributes

One feature that distinguishes Securus from previous approaches is that while encryption *and* fragmentation is used to protect data confidentiality, we do not enforce *true* fragmentation of the attributes on the SPs. In many scenarios, true fragmentation is an unnecessarily strict way of fulfilling confidentiality constraints. For instance, consider the example depicted in Figure 3. While *gender* and *ZIP* must not be stored by the same SP to satisfy the CC `[gender,ZIP]`, it is legitimate to store *age* at both SPs. True fragmentation would dictate that *age* may only be present at a single SP in plaintext.

ID	gender	age
1	m	30
2	m	50
3	m	40
4	f	20
5	f	40
6	f	40

Exemplary index table at SP1
(AP: `[gender, age]`)

ID	ZIP	age
1	76131	30
2	51362	50
3	76131	40
4	51362	20
5	76131	40
6	51362	40

Exemplary index table at SP2
(AP: `[ZIP, age]`)

Fig. 3. Exemplary index tables at two SPs. APs: `[ gender, age ]`, `[ ZIP, age ]`, CC: `[ gender, ZIP ]`

The example shown in Figure 3 also illustrates the advantages of storing attributes redundantly at multiple SPs. Consider the query `SELECT ID FROM index table 1 WHERE gender=f AND age=50`. This query would return zero records as no record matches the query. However, if the attributes were truly fragmented and *age* would not be part of index table 1 but of index table 2, the query would have to be reformulated to `SELECT ID FROM index table 1 WHERE gender=f`. Consequently, records 3-6 would be transmitted to the mediator and would then all be discarded when evaluating the second part of the query: `SELECT * FROM results WHERE age=50`. Thus, for this example, three records would have been transmitted unnecessarily due to true fragmentation.

Table 2. Time required to generate a mediator from policy profiles of various sizes

Number of policies/elements					Duration (s)	
Attr.	APs	ICs	CCs	SPs	mean	max
10	5	1	10	3	0.004	0.056
20	10	2	40	4	0.015	0.217
20	15	2	10	4	0.016	0.330
40	40	3	50	4	0.555	8.169
40	40	3	60	4	0.658	6.714
80	40	3	40	4	0.036	0.812
80	60	3	80	4	0.887	16.602
80	80	3	100	4	5.474	692.235

Furthermore, the example illustrates why Securus can provide hard performance guarantees for specified APs. Securus enforces that for each specified AP, an index table that holds all attributes of the AP exists. Thus, instead of having to evaluate parts of the query in the mediator, queries can be entirely executed by the SPs. As shown in the example above, true fragmentation can not guarantee that all returned records are part of the query's result. In fact, the number of unnecessarily transmitted records depends on the datasets structure. As the example shows, it is possible that, to answer a query that would not return any results at all, half of the dataset needs to be transmitted to the mediator first. Thus, client-side validation whether a record is really contained in the result or not can lead to a massive and potentially unpredictable performance overhead.

5.2 Policy Transformation Performance

To evaluate the policy transformation performance, we randomly generated policy profiles of different sizes and measured the time needed to transform the policy profiles into mediators. For each policy profile size, we generated 5000 policy profiles. We ran the Gurobi solver[2] on a commodity computer with 4GB RAM and a 2.93GhZ Dual Core CPU to conduct the measurements shown in Table 2. The results show that Securus scales well for reasonably sized scenarios. The maximum transformation time was less than 693 seconds even for datasets with 80 attributes, 80 APs and 100 CCs. This is feasible, as mediator generation only has to be performed once, initially.

5.3 Discussion

Securus allows the definition of APs that guarantee an efficient query execution of the according queries and CC that prevents SPs from viewing attribute combinations or even from inspecting single attribute values. Policy profiles that are inherently unsolvable exist. As a trivial example, consider a policy profile with

[2] `http://www.gurobi.com`

a CC $[a_1, a_2]$ and an AP $[a_1, a_2]$. The CC and the AP contradict each other: both attributes have to be present at an SP in plaintext or as deterministic substitutes to satisfy the AP, however, at least one attribute must not be stored as plaintext or as deterministic substitute to satisfy the CC. In future work, we will investigate how to support the user in resolving policy conflicts. In particular, the solution will tell the user *why* his policy profile is infeasible.

In terms of query constructs, Securus supports equality and range selections as well as aggregation. It can be easily extended to support further query constructs such as `LIKE` or `GROUP` operators. Just like the range and equality selections, these would have to be bound to a required representation (e.g., plaintext or deterministic substitute) at the SP.

We assume that the CCs, APs and ICs are known by the user before generating the mediator and are not subject to changes. However, CCs might change in reality, for instance, due to shifting legal requirements concerning privacy. APs might change due to the arising demand to efficiently execute queries that have not been addressed by existing APs. To account for changing policies, a new mediator has to be generated and the dataset has to be re-outsourced.

The proposed concept considers each SP as an honest-but-curious attacker. Unlike other approaches [17,9], we do not assume that the attacker is not capable of monitoring operations on the data or queries. We only assume that the attacker lacks applicable background knowledge on the executed operations and queries such as the frequency distribution of executed queries. In particular, using Securus an SP without background knowledge can not violate CCs by monitoring the effects of an insert operation on the data tables. In future work we plan to extend Securus to also address attackers *with* background knowledge on incoming queries.

6 Conclusion

We presented an outline of Securus, a framework that simplifies data outsourcing by allowing users to specify their requirements in terms of data confidentiality and queries that have to be efficiently executable. Based on this information a software adapter is generated that can be used to outsource the data compliantly to the specified requirements and that can execute the specified queries efficiently. We extended our previous publication of Securus [14] by highlighting the differences to other data outsourcing frameworks in this paper. One major difference constitutes that Securus makes use of both fragmentation and encryption techniques, but also stores attributes redundantly at multiple SPs for a more efficient query execution if this does not undermine the required data confidentiality. Furthermore, it was shown that Securus can provide hard performance *and* confidentiality guarantees. We altered the policy model of the originally published version of Securus by specifying ICs that make assertions on the assumed attacker's knowledge. In future work, we can utilize the strict semantics of ICs to pursuit one of our future research directions: enabling users to model the attacker's capabilities in a fine-grained manner. Furthermore, we aim to explore methods to resolve conflicting requirements and to adapt the approach to

consider fine-grained performance differences of both security mechanisms and database backends. Although Securus abstracts from confidentiality enhancing techniques, users are still required to specify which information needs protection. Enabling users to accurately define their company's individual confidentiality needs constitutes an interdisciplinary challenge that needs to be addressed not only in computer science but also in law and social sciences.

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: Proc. of CIDR (2005)
2. Armbrust, M., Fox, A., Griffith, R., Joseph, A., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., Zaharia, M.: Above the clouds: A Berkeley view of cloud computing. Technical report, Berkeley (2009)
3. Camenisch, J., Dubovitskaya, M., Lehmann, A., Neven, G., Paquin, C., Preiss, F.-S.: Concepts and languages for privacy-preserving attribute-based authentication. In: Fischer-Hübner, S., de Leeuw, E., Mitchell, C. (eds.) IDMAN 2013. IFIP AICT, vol. 396, pp. 34–52. Springer, Heidelberg (2013)
4. Ceselli, A., Damiani, E., De Capitani di Vimercati, S., Jajodia, S., Paraboschi, S., Samarati, P.: Modeling and assessing inference exposure in encrypted databases. ACM Transactions on Information and System Security (TISSEC) (2005)
5. 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 Transactions on Information and System Security (TISSEC) (2010)
6. Curtmola, R., Garay, J., Kamara, S., Ostrovsky, R.: Searchable symmetric encryption: improved definitions and efficient constructions. In: Proc. of the 13th ACM Conference on Computer and Communications Security (CCS). ACM (2006)
7. Damiani, E., De Capitani di Vimercati, S., Jajodia, S., Paraboschi, S., Samarati, P.: Balancing confidentiality and efficiency in untrusted relational DBMSs. In: Proc. of the ACM Conf. on Computer and Communications Security (CCS) (2003)
8. De Capitani di Vimercati, S., Foresti, S., Jajodia, S., Paraboschi, S., Samarati, P.: On information leakage by indexes over data fragments. In: Proc. of the 1st Int. Workshop on Privacy-Preserving Data Publication and Analysis (PrivDB) (2013)
9. Foresti, S.: Preserving Privacy in Data Outsourcing. Springer (2011)
10. Hacıgümüş, H., Iyer, B., Li, C., Mehrotra, S.: Executing SQL over Encrypted Data in the Database-Service-Provider Model. In: Proc. of SIGMOD (2002)
11. Hacıgümüş, H., Iyer, B., Mehrotra, S.: Efficient execution of aggregation queries over encrypted relational databases. In: Lee, Y., Li, J., Whang, K.-Y., Lee, D. (eds.) DASFAA 2004. LNCS, vol. 2973, pp. 125–136. Springer, Heidelberg (2004)
12. Harauz, J., Kaufman, L.M., Potter, B.: Data Security in the World of Cloud Computing. IEEE Security and Privacy (2009)
13. Jaatun, M.G., Bernsmed, K., Undheim, A.: Security sLAs – an idea whose time has come? In: Quirchmayr, G., Basl, J., You, I., Xu, L., Weippl, E. (eds.) CD-ARES 2012. LNCS, vol. 7465, pp. 123–130. Springer, Heidelberg (2012)
14. Jünemann, K., Köhler, J., Hartenstein, H.: Data outsourcing simplified: Generating data connectors from confidentiality and access policies. In: Proc. of the Workshop on Data-intensive Process Management in Large-Scale Sensor Systems (CCGrid-DPMSS) (2012)

15. Mykletun, E., Tsudik, G.: Aggregation queries in the database-as-a-service model. In: Data and Applications Security XX, pp. 89–103 (2006)
16. Pearson, S.: Toward accountability in the cloud. IEEE Internet Computing 15(4), 64–69 (2011)
17. Popa, R.A., Redfield, C., Zeldovich, N., Balakrishnan, H.: Cryptdb: protecting confidentiality with encrypted query processing. In: Proc. of the 23rd ACM Symp. on Operating Systems Principles (SOSP), pp. 85–100. ACM (2011)
18. Song, D.X., Wagner, D., Perrig, A.: Practical techniques for searches on encrypted data. In: Proc. of the IEEE Symposium on Security and Privacy (S&P) (2000)
19. Xiong, L., Goryczka, S., Sunderam, V.: Adaptive, secure, and scalable distributed data outsourcing: a vision paper. In: Proc. of the Workshop on Dynamic Distributed Data-intensive Applications, Programming Abstractions, and Systems (3DAPAS) (2011)