Transparent Access on Encrypted Data Distributed over Multiple Cloud Infrastructures

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

Adriano Enrico Scaruffi
Doxee SpA
Modena, Italy
ascaruffi@doxee.com

***Abstract*—Using cloud infrastructures to store and backup data is becoming a popular alternative that guarantees performance and scalability at reasonable prices. However, standard cloud solutions could raise some concerns about data confidentiality and dependency on a single provider. We aim to address these issues by using cloud storage of multiple cloud providers. Our solution ciphers, partitions and replicates data among multiple cloud architectures, thus augmenting availability and confidentiality, and avoiding lock-in of one cloud provider. The proposed model is implemented through open source software that leverages data storage offered by multiple providers. This prototype demonstrates the effectiveness of the geographically distributed architecture in several real case scenarios.**

Keywords- cloud; storage; encryption; file fystem; replication

I. INTRODUCTION

Cloud storage is an interesting alternative that allows users to leverage huge size disk spaces characterized by high availability and scalability at pay-per-use cost models. However, when companies outsource their information to the cloud, there are many concerns about data confidentiality and complete dependency on one cloud provider. Issues such as law restrictions [1], vendor lock-in and unavailability cases causing service interruptions and data losses (e.g., [2]) are limiting a widespread adoption of cloud storage solutions.

This paper proposes a novel architecture that aims to augment data resiliency and confidentiality, and to avoid possible lock-in related to one cloud provider. The idea is to implement a virtual file system where data are encrypted, replicated and disseminated among different cloud providers. In such a way, there is no dependence on one provider, and adopted encryption schemes are robust even against insider attacks and colluding providers. Moreover, we consider it important to provide users with a transparent encrypted access to such virtual file system. Thanks to the proposed standard file system interface, any application operating on files can leverage the proposed architecture without software modifications. In this paper, we demonstrate the efficacy of the proposed architecture by running a relational database on top of it.

Existing solutions [3]–[5] concerning data confidentiality, integrity and replication for untrusted storage services do not meet all requirements about encryption, replication and transparency. For example, data replication is not considered in [3]. Unlike our architecture based on the Infrastructure as a Service (IaaS) paradigm, the system described in [4] refers to the more sophisticated and expensive Storage as a Service paradigm. This scheme transparently provides customers with advanced techniques for elasticity, scalability and availability, but it requires the implementation and maintenance of dedicated drivers for each cloud storage API, thus causing additional cloud lock-in problems. The interesting solution proposed in [5] has two drawbacks: it is not quite transparent to the customer because it requires changes at the level of application logic; moreover, it is not resistant against colluding cloud providers.

The proposed architecture guarantees data confidentiality and integrity at rest, in motion and in use. To provide users with complete confidentiality of outsourced data we adopt encryption techniques and algorithms of proven security. Data are replicated in a multi-tenant architecture built over multiple cloud storage services. In this paper, we describe the overall model, the details of the architecture components, and the guidelines for its implementation.

The remaining part of this paper is structured as following. Section II analyzes other solutions related to our proposal. Section III describes the architectural model and the main requirements. Section IV reports the internal details of the proposed architecture and the main functionalities. Section V presents an example of a relational database that can leverage the proposed architecture. A summary of the results is reported in Section VI.

II. RELATED WORK

Data confidentiality on untrusted storage was initially guaranteed by encrypted file systems (e.g., [3], [6]) that allow a customer to encrypt all data stored in a cloud IaaS. However, these solutions do not allow to slice and to replicate data among several cloud providers as provided by previous architectures including that proposed in this paper.

Some academic and commercial proposals guaranteeing data confidentiality and integrity by using multi-tenant cloud services are recently appearing. The solutions most related to this paper are iDataGuard [4] and Depsky [5]

 ISBN: 978-1-61208-271-4

iDataGuard is a middleware that leverages the cloud Storage as a Service paradigm. This approach differentiates iDataGuard from our solution that is based on the standard IaaS paradigm. Cloud storage services can take advantage of several benefits with respect to IaaS, because they transparently provide customers with advanced API-based solutions for elasticity, scalability and availability. These techniques facilitate the low level implementation of iDataGuard, but they require the software implementation and maintenance of dedicated drivers for each specific cloud storage API. As a consequence, this solution limits portability and reduces the possibility of avoiding cloud provider lock-in. We should also observe that iDataGuard does not transparently replicate information among the cloud storage services, but data are managed by users as distinct storage units.

Depsky [5] proposes an interesting storage architecture that allows key-value access to data and guarantees data consistency also in the worst case of Byzantine faults. Depsky requires clients to access intermediate trusted components that provide key distribution by means of a Shamir secret sharing scheme [7]. This does not guarantee data confidentiality in the case of colluding cloud providers. Another problem is that applications based on Depsky require changes at the software level, because this architecture comes with a non-standard interface for data management.

Other papers (e.g., [8], [9]) aiming to guarantee confidentiality of information stored in untrusted storage servers can avoid data encryption. For example, they guarantee k-anonymity [10] by splitting sensitive data among multiple subsets, each managed by an independent cloud provider. Since data are not encrypted, each cloud can obtain some information on a portion of data. Moreover, such techniques require a complete awareness of the underlying data structure, that are against our main design requirement that the proposed solution must be transparent to the applications. In order to guarantee data confidentiality in the cloud database paradigm, full homomorphic encryption [11] is described as the final solution for single client computing scenarios [12]. In practice this approach is not yet feasible because of the prohibitive computational costs on possible operations.

A different set of proposals are oriented to cloud database services that differ from the architecture proposed in this paper because of the logical software level, and lack of transparency and portability. For example, some DBMS engines provide users with advanced proprietary techniques to encrypt data at storage level (e.g., Transparent Data Encryption (TDE) [13]). These features can replace the encryption layer of the proposed architectures, and can improve performances thanks to data caching and selective blocks retrieval. However TDE implementation is related to some specific DBMS, and many database services do not propose any similar solution. On the other hand, we remark that the proposed architecture aims to be transparent of any specific DBMS and cloud-related solution. Cloud database as a service (e.g., [14]–[17]) is an interesting alternative to support database in cloud infrastructures. They have the advantage of executing database SQL computations directly on the cloud infrastructure and to leverage intrinsic scalability and reliability of a cloud provider. However, there are no proposals that are oriented to federate databases among multiple cloud providers.

III. Model overview

An architecture guaranteeing maximum availability and security on untrusted storage services should satisfy the following main objectives.

- Confidentiality must be guaranteed for data at rest, in motion an in use without any risk of information leakage due to cloud insiders and collusive providers.
- Service availability must not depend on one cloud provider.
- The proposed architectures should be transparent to the supported applications in the broadest sense, that is, no modifications must be required at the application level.

To satisfy all the previous objectives we propose the architectural model that is represented in Figure 1.

Let us consider an application that executes some operations requiring accesses to data storage. This is a *plain data* scenario where the application does not provide any solutions to guarantee data confidentiality. The application executes virtual data operations on a file system, as if it were on local storage. As transparency is one of the main objectives of the proposed architecture, our solution adopts a standard file system interface that guarantees the required level of transparency. In practice, data are not stored in local devices nor in a local network environment as it is usually done in private data centers. Instead, all data are stored in multiple cloud infrastructures. Other main logical components of the proposed architecture are the *data encryptor*, the *data slicer* and the *data replication* modules. The combination of all of them guarantees confidentiality, availability and resiliency of data managed by the application.

To give a high-level description of the architecture model, we initially identify a trusted area and untrusted areas. The trusted area is under the direct control of the data owner, and can be accessed by only trusted third-party subjects. Plain data must never access the untrusted area before being encrypted. All security policies and decryption keys must be managed by trusted parties.

Each application executes operations on plain data and does not require any software modifications in order to guarantee the correct execution of the security techniques that our solution applies. It is the proposed architecture that provides applications with a standard file system interface allowing them to manage data as in local storage devices, although data are really stored on several Infrastructures as

 ISBN: 978-1-61208-271-4

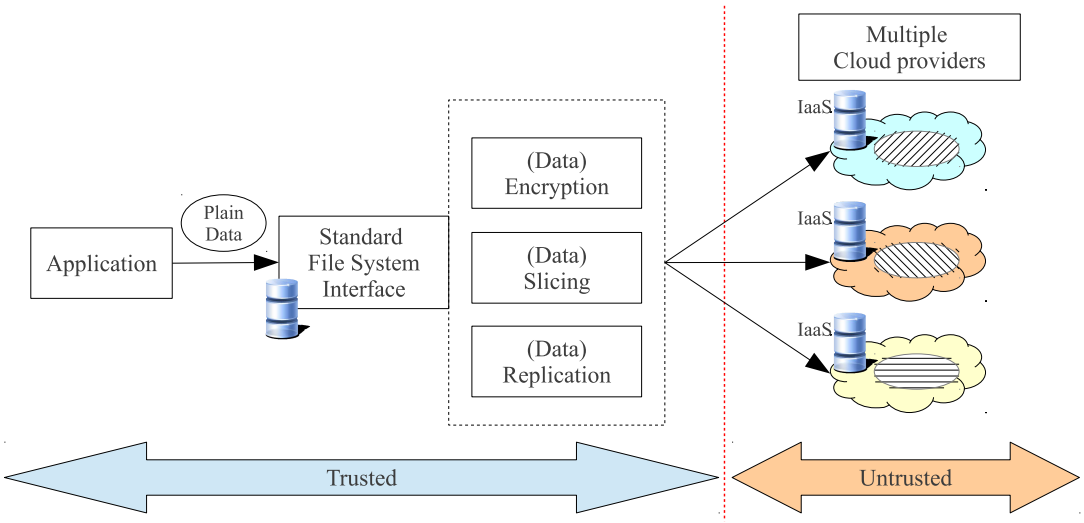

Figure 1. Architectural model.

a Service (IaaS) resources that are under the direct control of multiple cloud providers.

Cloud IaaS is not the only choice to use cloud storage because cloud providers offer also cloud storage as a service solutions through high level APIs facilitating data management. Our choice of preferring cloud IaaS instead of the cloud storage as a service paradigm was motivated by the following three reasons.

1) The IaaS paradigm allows us to directly manage virtual machines and disk resources that are standard; consequently, we can install and configure the best solutions to satisfy the architectural requirements of transparency and data confidentiality.
2) Cloud storage as a service requires data to be managed through proprietary APIs. This may cause some forms of cloud lock-in and may limit the portability of the solutions.
3) Cloud storage as a service can transparently provide advanced replication techniques to guarantee advanced resiliency. However, these benefits can be achieved also through the proposed architecture without any additional reliance on non-standard cloud services.

In our proposal, plain data received from an application are subjected to two types of manipulations:

- encryption to guarantee information confidentiality;
- distribution over multiple cloud infrastructures to increase availability and avoid dependency on one provider.

Slicing and replication reinforce security in the worst case scenario of collusion between a cloud provider and an internal (theoretically trusted) subject, because it prevents a cloud provider from accessing the whole data set. Moreover, they are useful to increase performance because they allow the parallelization of some data operations, and reduce space overhead caused by replication.

In the following Section IV we describe the details of the architecture and outline its implementation.

IV. ARCHITECTURE

The paper proposes a novel architecture that allows clients to leverage remote storage of multiple cloud providers. While internally managed infrastructures allow data owner to directly control data security policies, the cloud paradigm has the advantage to reduce costs and augment scalability, availability and resiliency. On the other hand, it opens user concerns in terms of data confidentiality and dependency on one provider.

We describe the implementation of the architectural model described in Figure 1 by referring to the architecture represented in Figure 2. A possible alternative based on a broker implementation is outlined in Figure 5.

Users applications transparently execute data operations on a logical file system, that is implemented by the interface layer of the proposed solution. Data replication strategies over multiple cloud storage servers are implemented by the **secure data management** (SDM) component. It guarantees that all data are stored in the infrastructures of at least two cloud providers (*high reliability*), and no provider owns all data (*high confidentiality*).

The Secure Data Management (SDM) represents the core of the proposed architecture that is typically implemented on an intermediate server. This proxy executes encryption and data distribution schemes over all application data, making use of multiple cloud providers to store encrypted encrypted data. The main modules of the SDM component are represented in Figure 3 and described below.

 ISBN: 978-1-61208-271-4

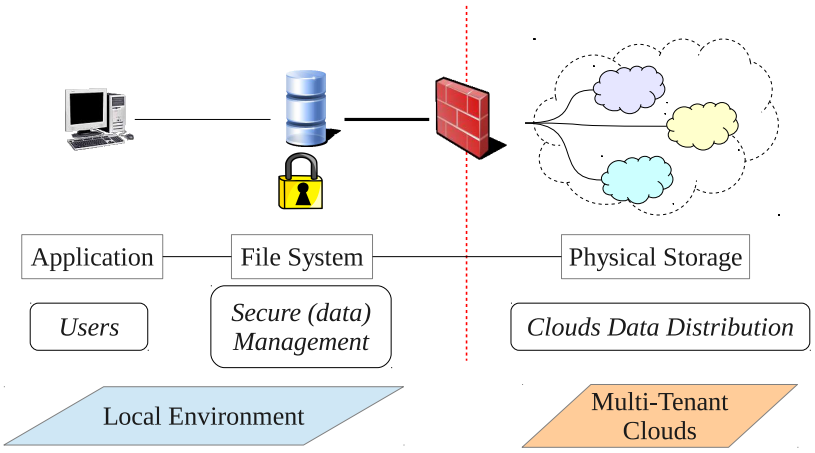

Figure 2. Architecture based on clients and cloud providers.

A. The file system interface implements a logical standard layer to the client applications.
B. The cache manager uses local storage to cache previously accessed data.
C. The disk encryptor implements well known encryption algorithms, such as AES [18], guaranteeing data confidentiality.
D. The distributed file system operates slice and replication policies on user data over multiple cloud providers. The possible alternatives and details are presented below.
E. The virtual private network guarantees confidentiality on untrusted channels of communication by encrypting all data in transit and authentication schemes for the cloud services.

Plain data of the user applications flow through the software modules of the intermediate proxy that fulfills all main requirements described in Section III. The most visible interface for the client applications is a **logical file system**. When stored data are accessed or modified by a client application, the logical file system searches for a hit in its local cache. If no match is found, then the request is forwarded to the underlying SDM modules. The performance benefits of caching strategies in geographically remote cloud storages is of paramount importance as evidenced in [19].

The **encryption module** transparently encrypts all data received from the logical file system. We use a software block mapping device that maintains a unique correspondence between an underlying encrypted storage and a logical interface to an unencrypted virtual device. In this version of our architecture, we use Dm-Crypt [20] that is a valid block mapper solution integrated in modern Linux kernels.

The underlying encrypted data are stored in the **distributed file system** (DFS) that replicates data among multiple cloud services. Since cloud IaaS services are commonly accessed by a global IP address as a remote host, any DFS

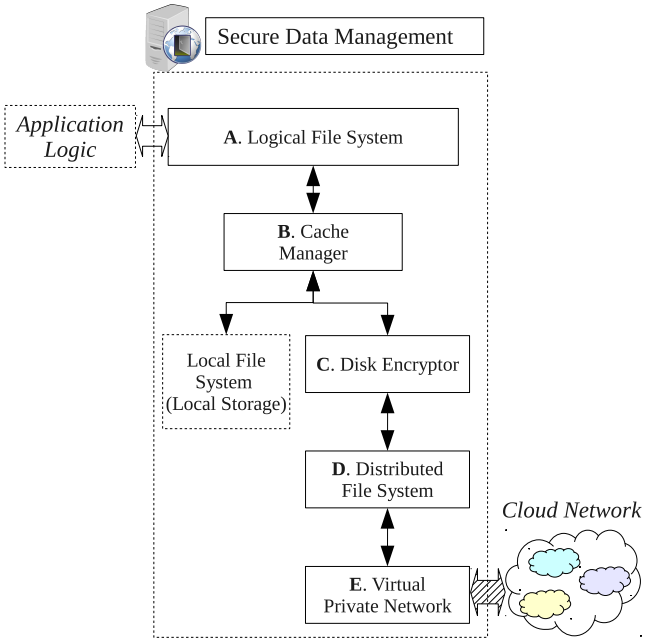

Figure 3. Software modules of the Secure Data Management (SDM) component.

can be used without any modifications. GlusterFS [21] is the chosen DFS satisfying our requirements. It installs and configures software components at the local side (clients) and at the remote cloud side (servers). Different file systems can operate different slicing and replication policies by using data at different system levels, such as blocks, files, volumes. The proposed architecture does not restrict the use of any specific policy, but our implementation choice (GlusterFS) distributes data at the file level, and guarantees integrity of data though hashing algorithms.

A **virtual private network** (VPN) adds a further level of confidentiality. It is not strictly necessary and it can be

 ISBN: 978-1-61208-271-4

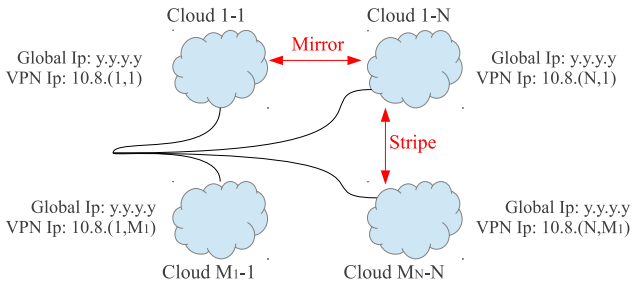

Figure 4. Network configuration of the multiple cloud storages.

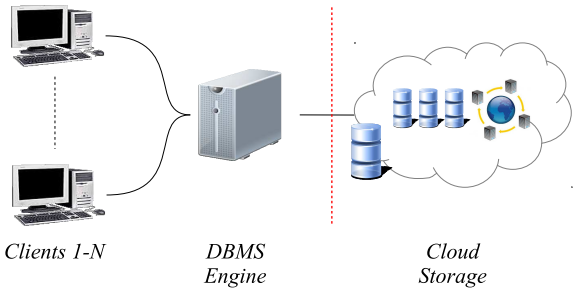

Figure 6. Example of a DBMS deployed over a cloud infrastructure.

avoided when performance becomes an issue. Through the VPN we can also configure the distributed cloud storages as if they were in a local network. Administrators can configure the network of cloud services by using common secure network mechanisms, such as firewalls, subnets and virtual LANs. OpenVPN [22], which is our choice for the present version of the software, is deployed at the local (server) and cloud sides (clients).

Benefits given by the use of the distributed file system and the virtual private network are represented in Figure 4. Each cloud IaaS is identified by the global IP address, and the VPN allows the configuration of a virtual network among the cloud services and the host that executes our software solution. Hence, we can associate each cloud service with a local network address. In the represented scheme, G cloud storage are grouped in N sets. Each group of clouds n has M_n members, such that $\sum_{n=1}^{N} M_n = G$. Clouds of the same set share the same subnet in the VPN network and are configured on a striping replication configuration. The striping configuration avoids that one cloud provider can manage the entire data set. The different subnets are configured in a mirror replication to increase availability and to break dependency on a single cloud provider. We notice that the possibility of using groups of different sizes is allowed only if the distributed file system can administrate data striping independently for each replicated data. Using groups of different sizes can be useful to balance data among infrastructures with different resource capabilities (we depend on network bandwidth and storage) and respective costs.

It is also important to specify that the proposed architectural solution can be deployed through a third party broker that implements the SDM components. This alternative has the great advantage of avoiding that a customer company must manage the complexity of the SDM, and additional secure infrastructures. This alternative is represented in Figure 5 and outlined below. In such a case, clients communicate with the broker proxy gateway through standard Internet protocols. The broker can be a different company that has direct contacts and contracts with multiple cloud providers. It implements the entire virtual file system and, thanks to an intermediate proxy server, it provides a storage service to the users. The trade-off of this alternative configuration should be clear: the customer can benefit from a simplified interface that avoids any implementation complexity; on the other hand, the broker must be a trusted subject.

V. USAGE SCENARIO

A relational database (DBMS) is a typical application that can take advantage of the proposed architecture. We initially consider an existing scenario, represented in Figure 6, where the database engine is deployed in a local environment, while the data storage is moved to a storage service related to a cloud provider. In a similar architecture, the data owner can take advantage of scalability and adequate resilience, but it does not have any guarantees about confidentiality of data stored to an external cloud service. Moreover, availability and data accessibility depend on one cloud provider that must be trusted by the data owner. While this scenario could be acceptable for some private customers using a cloud storage to backup non-critical information, most companies require additional guarantees about confidentiality and availability before outsourcing data.

Thanks to the proposed architecture we can guarantee that data stored in the cloud is confidential, and that a cloud provider cannot prevent the data owner from accessing its data. We show the configuration related to the broker-less solution in Figure 7.

Clients execute database operations to the local DBMS engine that is connected to the interface of the secure file system to manage data to/from the cloud storages. In this example, we use four cloud providers, where two groups of two clouds are internally organized in a striping configuration, and two groups are configured in a mirroring configuration. Each cloud provides us with an infrastructure as a service paradigm (IaaS), where we can install the distributed file system servers and the virtual private network clients. The encryption layer encrypts all data that are sent by the DBMS, and decrypts all requested data by imposing the database storage in the virtual space created by the device mapper. The distributed file system guarantees that no cloud providers can store the entire data set, because each of them

 ISBN: 978-1-61208-271-4

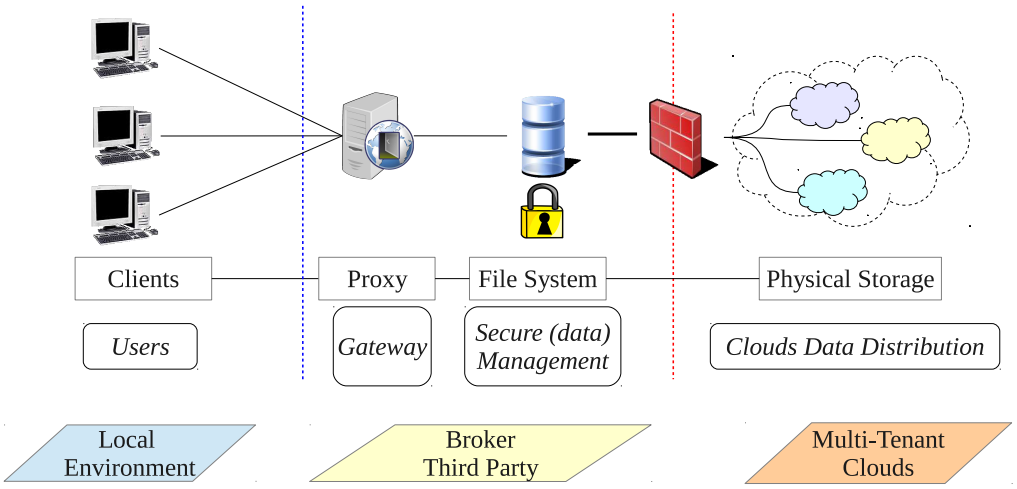

Figure 5. Architecture based on clients, third party broker and cloud providers.

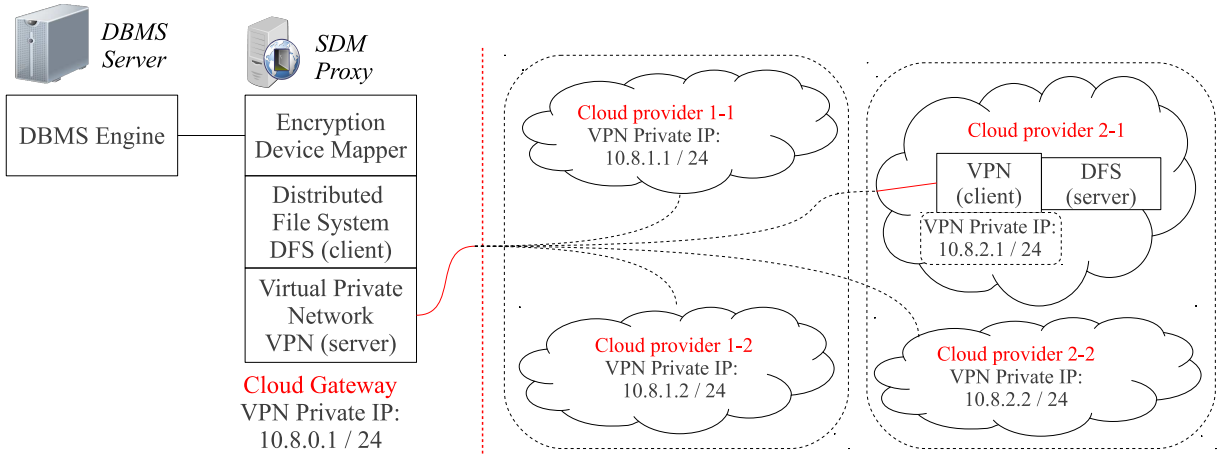

Figure 7. Example of a DBMS deployed over multiple cloud infrastructures (local manager).

has at most half of the entire data set, and all data are stored in at least two cloud providers.

The virtual private network allows us to guarantee security over the access to the clouds, and to configure the replication as if it were in a local area network. As described in Section IV, the clouds that are configured in a striping distribution share the same subnet.

It is important to observe that all tools of the deployed DBMS engine can be used as in a full local environment. Users access policies can be managed as in a standard unencrypted database architecture, because encrypted data are transparently managed by the DBMS engine through the file system interface of the proposed solution. We highlight that this configuration performance can benefit of the DBMS engine caching policies, in addition to the caching mechanisms provided by our architecture (see Section IV).

In this example, the DBMS engine is implemented in PostgreSQL [23], that is a well-known open-source relational database. It can be deployed in the proposed architecture because it stores data in a standard directory that can be redirected to the file system interface of the proposed solution. Moreover, it allows us to leverage caching policies that are aware of the structure of the database and of the queries.

VI. Conclusions

This paper proposes a novel architecture to leverage multiple cloud storage services while guaranteeing data confidentiality and avoiding customer dependency on one cloud provider.

Data confidentiality is guaranteed by means of classical encryption schemes; data are replicated among several cloud providers through striping and mirroring techniques. Striping increases performance and data protection, because it prevents that one cloud provider stores the whole data set. The proposed architecture is transparent to the application

 ISBN: 978-1-61208-271-4

layer, as it provides the client with a standard file system interface.

We demonstrate how the proposed architecture can be implemented through open source software components. Moreover, we show that it is suitable to support any kind of applications working on storage service; in this paper, we consider the complex case of a relational database, but other applications are supported as well. This work was focused on the feasibility of the proposal, while performance tests for different workload models and network latencies represent an ongoing work.

REFERENCES

[1] W. Jansen and T. Grance, "Guidelines on security and privacy in public cloud computing," NIST special publication, pp. 800–144, 2011.

[2] The New York Times, "Amazon's trouble raises cloud computing doubts," http://www.nytimes.com/2011/04/23/technology/23cloud.html, March 2013.

[3] M. Kallahalla, E. Riedel, R. Swaminathan, Q. Wang, and K. Fu, "Plutus: Scalable secure file sharing on untrusted storage," in Proceedings of the 2nd USENIX Conference on File and Storage Technologies, vol. 42, 2003, pp. 29–42.

[4] R. Jammalamadaka, R. Gamboni, S. Mehrotra, K. Seamons, and N. Venkatasubramanian, "Idataguard: middleware providing a secure network drive interface to untrusted internet data storage," in Proceedings of the 11th international conference on Extending database technology: Advances in database technology. ACM, 2008.

[5] A. Bessani, M. Correia, B. Quaresma, F. André, and P. Sousa, "Depsky: dependable and secure storage in a cloud-of-clouds," in Proceedings of the 6th conference on Computer systems. ACM, 2011, pp. 31–46.

[6] E. Zadok, I. Badulescu, and A. Shender, "Cryptfs: A stackable vnode level encryption file system," Citeseer, Tech. Rep., 1998.

[7] A. Shamir, "How to share a secret," Communications of the ACM, vol. 22, no. 11, pp. 612–613, 1979.

[8] V. Ciriani, S. D. C. Di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, and P. Samarati, "Keep a few: Outsourcing data while maintaining confidentiality," in Proceedings of the 14th European Symposium on Research in Computer Security. Springer, 2009, pp. 440–455.

[9] P. Samarati, "Protecting respondents identities in microdata release," IEEE Transactions on Knowledge and Data Engineering, vol. 13, no. 6, pp. 1010–1027, 2001.

[10] L. Sweeney, "k-anonymity: A model for protecting privacy," International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, vol. 10, no. 05, pp. 557–570, 2002.

[11] C. Gentry, "A fully homomorphic encryption scheme," Ph.D. dissertation, Stanford University, 2009.

[12] M. Van Dijk and A. Juels, "On the impossibility of cryptography alone for privacy-preserving cloud computing," in Proceedings of the 5th USENIX conference on Hot topics in security. USENIX Association, 2010, pp. 1–8.

[13] Oracle corporation, "Oracle advanced security," http://www.oracle.com/technetwork/database/options/advanced-security, March 2013.

[14] H. Hacigumus, B. Iyer, and S. Mehrotra, "Providing database as a service," in Proceedings. of the 18th International Conference on Data Engineering. IEEE, 2002, pp. 29–38.

[15] L. Ferretti, M. Colajanni, and M. Marchetti, "Supporting security and consistency for cloud database," in Proceedings of the 4th International Symposium on Cyberspace Safety and Security. Springer, 2012, pp. 179–193.

[16] H. Hacigümüş, B. Iyer, C. Li, and S. Mehrotra, "Executing sql over encrypted data in the database-service-provider model," in Proceedings of the 2002 ACM SIGMOD international conference on Management of data. ACM, 2002, pp. 216–227.

[17] R. A. Popa, C. Redfield, N. Zeldovich, and H. Balakrishnan, "Cryptdb: protecting confidentiality with encrypted query processing," in Proceedings of the 23rd ACM Symposium on Operating Systems Principles. ACM, 2011, pp. 85–100.

[18] J. Daemen and V. Rijmen, The design of Rijndael: AES-the advanced encryption standard. Springer, 2002.

[19] M. Vrable, S. Savage, and G. M. Voelker, "Bluesky: A cloud-backed file system for the enterprise," in Proceedings of FAST, 2012, pp. 237–250.

[20] Dm-Crypt, "Linux kernel device-mapper crypto target," http://code.google.com/p/cryptsetup/wiki/DMCrypt, March 2013.

[21] GlusterFS, "Open source, distributed file system," http://www.gluster.org, March 2013.

[22] OpenVPN, "Open source vpn," http://openvpn.net, March 2013.

[23] The PostgreSQL Global Development Group, "Postgresql," http://code.google.com/p/cryptsetup/wiki/DMCrypt, March 2013.

 ISBN: 978-1-61208-271-4