# CLOUD DATA PATTERNS FOR CONFIDENTIALITY

Steve Strauch[1], Uwe Breitenbuecher[1], Oliver Kopp[1], Frank Leymann[1] and Tobias Unger[1,2]
[1]*Institute of Architecture of Application Systems, University of Stuttgart, Stuttgart, Germany*
[2]*GRIDSOLUT GmbH + Co. KG, Stuttgart, Germany*

Keywords: Patterns, Confidentiality, Database Layer, Migration, Distributed Application Architecture, Cloud Data Store.

Abstract: Cloud computing enables cost-effective, self-service, and elastic hosting of applications in the Cloud. Applications may be partially or completely moved to the Cloud. When hosting or moving the database layer to the Cloud, challenges such as avoidance of disclosure of critical data have to be faced. The main challenges are handling different levels of confidentiality and satisfying security and privacy requirements. We provide reusable solutions in the form of patterns.

## 1 INTRODUCTION

Cloud computing has started to be applied in industry due to the advantage of reducing capital expenditure and transforming it into operational costs (Armbrust et al., 2009). Therefore, applications are directly built using Cloud service technology or existing applications are migrated to the Cloud.

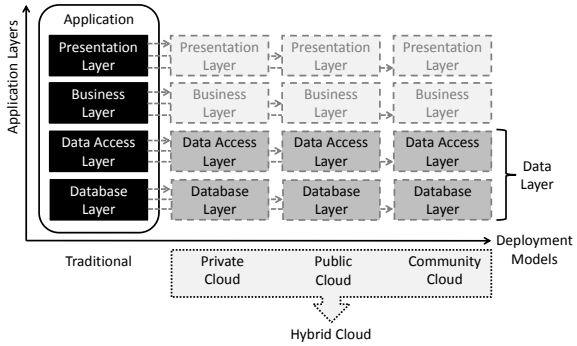



Figure 1: Overview of Cloud Deployment Models and Application Layers.

Applications are typically built using a three layer architecture model consisting of a presentation layer, a business logic layer, and a data layer (Dunkel et al., 2008). The data layer is responsible for data storage and is in turn subdivided into data access layer (DAL) and database layer (DBL). The DAL is an abstraction layer encapsulating the data access functionality. The DBL is responsible for data persistence and data manipulation. The subdivision of the data layer finally leads to a four layer application architecture (Figure 1). To take advantage of Cloud computing, an application may be moved to the Cloud or designed from the beginning to use Cloud technologies. Figure 1 shows all potential possibilities for the distribution of an application consisting of four layers. The traditional application not using any Cloud technology is shown on the left. Each layer can be hosted using different Cloud deployment models. Available Cloud deployment models are: Private Cloud, Public Cloud, Community Cloud, and Hybrid Cloud (Mell and Grance, 2009). In this context, "on-premise" denotes that the Cloud infrastructure is hosted inside the company and "off-premise" denotes that it is hosted outside the company. Figure 1 will be reused for providing a sketch for each Cloud Data Pattern, illustrating the solution.

This paper deals with hosting or moving the database layer to the Cloud and the related challenges regarding data confidentiality to be faced. The contribution of the paper is the identification of these challenges and the description of Cloud Data Patterns to face these challenges. A *Cloud Data Pattern* describes a reusable and implementation technology-independent solution for a challenge related to the data layer of an application in the Cloud for a specific context. Cloud Data Patterns address both the migration of a data layer hosted traditionally to the Cloud as well as enabling access to the data layer in the Cloud. "Traditionally" denotes not using any Cloud technology. *Confidentiality Patterns* provide solutions for avoiding disclosure of confidential data. Confidentiality includes security and privacy. In the context of confidentiality we consider the data to be kept private and

Strauch S., Breitenbuecher U., Kopp O., Leymann F. and Unger T..
CLOUD DATA PATTERNS FOR CONFIDENTIALITY.
DOI: 10.5220/0003893903870394
In *Proceedings of the 2nd International Conference on Cloud Computing and Services Science* (CLOSER-2012), pages 387-394
ISBN: 978-989-8565-05-1

secure as “critical data”. For instance, critical data are business secrets of companies, personal data, and health care data. Critical data can be categorized into different confidentiality levels such as “NATO Secret” or “NATO Confidential” as described in the security classification in the security regulations of the North Atlantic Treaty Organization (NATO) (Department of Defense Security Institute, 1993).

To provide a set of Cloud Data Patterns as reference work, the description of each pattern has to be self-contained. Thus, the concrete characteristics of the patterns presented in Section 2 overlap. Section 3 presents related work in the field of patterns. Finally, Section 4 concludes and presents an outlook on future work.

# 2 CONFIDENTIALITY PATTERNS

This section introduces five new Cloud Data Patterns for confidentiality. As the data is not always categorized or might be categorized using different confidentiality categorizations we present two patterns for handling these challenges: The Confidentiality Level Data Aggregator (Section 2.1) provides one confidentiality level for data from different sources with potentially different confidential categorizations on different scales. The Confidentiality Level Data Splitter (Section 2.2) provides different confidentiality levels for data originally categorized into one confidentiality level. Both the aggregation and splitting have to be configured, e. g., using parameterization or configuration files.

To protect the confidential data on the one hand and to benefit from Cloud computing on the other hand, we introduce three different patterns: filtering, pseudonymization, and anonymization. The Filter of Critical Data (Section 2.3) ensures that no confidential data is disclosed to the public. The Pseudonymizer of Critical Data (Section 2.4) implements pseudonymization. Pseudonymization is a technique to provide a masked version of the data to the public while keeping the relation to the non-masked data (Federal Ministry of Justice, 1990). That enables processing of non-masked data in the private environment. The Anonymizer of Critical Data (Section 2.5) implements anonymization. Anonymization is a technique to provide a reduced version of the critical data to the public while it is impossible to relate the reduced version to the critical data. A discussion on the composition of Confidentiality Patterns is provided in Section 2.6.

## 2.1 Confidentiality Level Data Aggregator

*Context*. The data formerly stored in one traditionally hosted data store is separated according to the different confidentiality levels and stored in different locations. The business layer is separated into the traditionally hosted part processing the critical data and the business layer hosted in the public Cloud processing the non-critical data. As the application accesses data from several data sources, the different confidentiality levels of the data items have to be aggregated to one common confidentiality level. This builds the basis for avoiding disclosure of critical data by passing it to the public Cloud.

*Challenge*. How can data of different confidentiality levels from different data sources be aggregated to one common confidentiality level?

*Forces*. The data might be categorized into confidentiality levels from different scales. Thus, aggregation by normalizing scales is not always possible.

*Solution*. An aggregator is placed within the Cloud infrastructure of the Cloud data storage with the highest confidentiality level. The aggregator retrieves data from all Cloud data stores. Thus, it especially processes data with the highest confidentiality level, which may not be placed where data with a lower level resides. As a consequence, it has to be placed in a location where the demands of the highest confidentiality level are fulfilled.

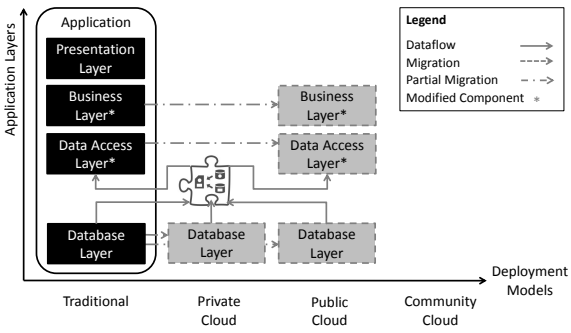



Figure 2: Sketch of Confidentiality Level Data Aggregator.

*Sidebars*. The data stored in the different data stores might be configured with different security levels from different security domains. The aggregator outputs data from one security domain only. The aggregator has to be configured accordingly and a mapping between different confidentiality levels from potentially different security or privacy domains has to be provided. Finally, a procedure for determining the final output level of the data combined by the aggregation has to be given. Usually, it is the highest con-

fidentiality level. Both parts of the data access layer hosted traditionally and in the public Cloud have to be reconfigured to access the database layer only through the confidentiality level data aggregator. It is assumed that the data stores themselves have appropriate access right configurations.

*Results*. For each request, several data stores are queried. The data is routed through the aggregator, which annotates the data with the appropriate confidentiality level valid for the aggregated data set.

*Example*. The database layer of an application built on Oracle Corporation MySQL (Oracle Corporation, 2011), version 5.1 is split into an Amazon Virtual Private Cloud (Amazon VPC (Amazon.com, Inc., 2011b)) data store hosting an AMI and Amazon Relational Database Service (Amazon RDS (Amazon.com, Inc., 2011a)). The AMI runs MySQL 5.1 relational database on OpenSolaris. Regarding Amazon RDS, a MySQL DB instance is chosen. Thus, the database functionality remains the same. The data stored in the Amazon VPC is annotated with "NATO Confidential" and the data stored in the Amazon RDS is annotated with "NATO Restricted" (Department of Defense Security Institute, 1993). The logic implemented in the business layer is split into the one processing critical and the one processing non-critical data. The data access layer is also split and configured to operate on the confidentiality level data aggregator. The aggregator is fetching the data from the two different data sources. In case data is returned from the Amazon VPC, the result set is annotated with "NATO Confidential", otherwise "NATO Restricted" is used. On its own, this annotation does not prevent the disclosure of data. For instance, the disclosure itself may be prevented in combination with the filter of critical data pattern.

*Next*. In case the stored data should be updated, the Confidentiality Level Data Splitter has to be considered.

## 2.2 Confidentiality Level Data Splitter

*Context*. The data formerly stored in one traditionally hosted data store is separated according to data stores with different confidentiality levels. As the application writes data to several data stores, the data has to be categorized and split according to the different confidentiality levels. This builds the basis for avoiding disclosure of critical data when storing it in the public Cloud.

*Challenge*. How can data of one common confidentiality level be categorized and split into separate data parts belonging to different confidentiality levels?

*Forces*. The data has to be annotated and split manually, in case the confidentiality level data splitter is not used.

*Solution*. A splitter is placed within the Cloud infrastructure of the data access layer. Thus, additional data movement, network traffic, and load can be minimized. The splitter writes data to all Cloud data stores. As the splitter processes data with the highest confidentiality level, it has to be placed in a location where the demands of the highest confidentiality level are fulfilled.

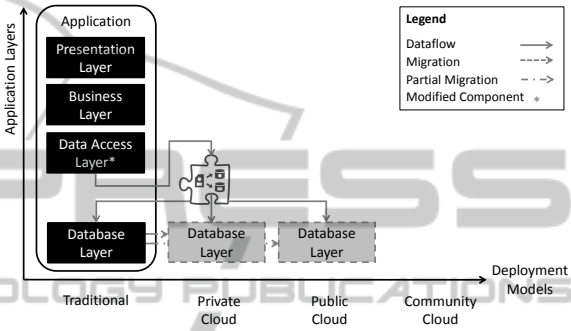



Figure 3: Sketch of Confidentiality Level Data Splitter.

*Sidebars*. The data to be stored in the different data stores might have to be separated and categorized into different confidentiality levels from one security or privacy domain. The splitter gets input data of a common security or privacy domain and outputs a separation of this data according to the particular security or privacy levels. The splitter has to be configured for the mapping from one common confidentiality level to different confidentiality levels from the same security or privacy domain. The configuration also includes a determination of the data store for each level. It is assumed that the data stores themselves have appropriate access right configurations. The data access layer has to be configured to write data exclusively through the Confidentiality Level Data Splitter to the database layer. In order to avoid disclosure of critical data, the data access layer has to be placed in a location where the demands of highest confidentiality level are fulfilled. Usually, this is the private Cloud.

*Results*. For each data write, several data stores are used. The data is routed through the splitter, which categorizes and separates the data of one common confidentiality level into disjoint data sets of different confidentiality levels. The data sets are stored in the appropriate data store based on the corresponding security level of each data set.

*Example*. The database layer of an application used by the NATO for management of data of different security levels consists of three data stores hosted

using different deployment models. The data store hosted traditionally is built on Oracle Corporation MySQL, version 5.1. The second and third parts using Amazon VPC Cloud data store hosting an AMI and Amazon RDS. The AMI runs MySQL 5.1 relational database on OpenSolaris. Regarding Amazon RDS, a MySQL DB instance is chosen. Thus, the database functionality remains the same. Now several tera bytes of new data sets have to be imported sequentially in the data management system. This data is neither categorized nor annotated regarding the classification levels defined by the NATO. Therefore, the splitter is used and configured as follows: All personal information has to be categorized as "NATO Secret" and stored in the Amazon VPC Cloud data store. All financial information have to be annotated with "NATO Confidential" and stored in the traditionally hosted part of the database layer. All other data is categorizes as "NATO Restricted" and stored in Amazon RDS.

*Next*. In case the stored data should be retrieved, the Confidentiality Data Level Aggregator has to be considered.

## 2.3 Filter of Critical Data

*Context*. The private Cloud data store contains critical and non-critical data. To prevent disclosure of the critical data it has to be enforced that the critical data does not leave the private Cloud. The logic implemented in the business layer is split into the one processing critical and the one processing non-critical data. The party implementing or hosting the processing of the non-critical data may not be trusted.

*Challenge*. How can data-access rights be kept when moving the database layer into the private Cloud and a part of the business layer as well as a part of the data access layer into the public Cloud?

*Forces*. As the business layer in the public Cloud needs to have access to the non-critical data, it is no option to completely forbid access to the database from the public Cloud. As the business layer hosted traditionally might need to have access to the complete data, the distribution of critical and non-critical data into different databases would require adjustments of the corresponding business logic and the data access layer hosted traditionally.

*Solution*. A Filter of Critical Data is placed within the Cloud infrastructure of the private Cloud data store. All requests to the private Cloud data store have to be directed to the filter. The private Cloud data store is only reachable by the filter of critical data. Requests for critical data originating off-premise are denied by the filter.

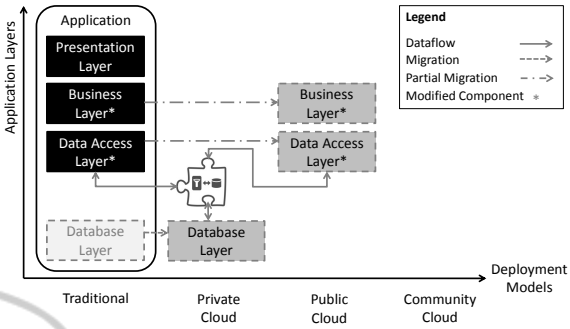



Figure 4: Sketch of Filter of Critical Data.

*Sidebars*. The data contained in the private Cloud data store has to be categorized into critical and non-critical data, e. g., by applying the security classification of the NATO (Department of Defense Security Institute, 1993) and annotating the data accordingly.

Both parts of the data access layer hosted traditionally and in the public Cloud have to be reconfigured to access the database layer only through the filter of critical data. We denote the required reconfigurations as "Data Access Layer*" in Figure 4. Moreover, the business layer hosted in the public Cloud has to be enabled to deal with errors occurring when data access to critical data is not granted, denoted by "Business Layer*" in Figure 4.

The method for filtering, the respective granularity, and the information to be filtered have to be configured, e. g., by parameterization.

The application in the public Cloud has to be aware that the data returned might be incomplete. Thus, the application design has to consider different availability of data. For instance, a function "write letters to all costumers" cannot be put in the public Cloud if the customer data must not be disclosed.

As the data access exclusively happens through the Filter of Critical Data this pattern decreases performance. Therefore, it is not applicable for distributed applications with high data traffic. Besides, it has to be avoided that the implementation of the filter becomes a single point of failure. To overcome that limitation, it could be horizontally scaled on demand, for instance.

*Results*. Assuming the data transfer from and to the private database layer happens exclusively through the Filter of Critical Data, only non-critical data is passed to the public Cloud and critical data remains private.

*Example*. The database layer of a task management application of a car manufacturing company built on Oracle Corporation MySQL (Oracle Corpo-

ration, 2011), version 5.1 is moved to Amazon Virtual Private Cloud (Amazon VPC (Amazon.com, Inc., 2011b)) choosing an Amazon Machine Image (AMI) running MySQL 5.1 relational database on OpenSolaris. Thus, the database functionality remains the same. The database contains (among others) tasks directly related to company internal research. These tasks are only allowed to be executed if the employee requesting the task details is currently located within the research technology center (cf. "Location Dependent Task Visibility and Access" (Unger and Bauer, 2008)). In case the request for such a task originates from outside the research technology center the access is denied by providing an error message. The location-based data filtering is IP-based and the internal IPs assigned to VPN connections are considered external.

*Next*. In case denying of access to critical data is not acceptable, the Anonymizer of Critical Data or Pseudonymizer of Critical Data have to be considered.

## 2.4 Pseudonymizer of Critical Data

*Context*. The private Cloud data store contains critical and non-critical data. The business layer is partially moved to the public Cloud and needs access to data. The logic implemented in the business layer is split into one requiring critical data and one, where critical data in pseudonymized form is sufficient for processing. This modification of the original business layer is denoted by "Business Layer*" in Figure 5. The party hosting the processing for which critical data in pseudonymized form is sufficient may not be trusted. Furthermore, passing critical data may be restricted by compliance regulations. It is required to relate the pseudonymized processing results from the public business layer to the critical data.

*Challenge*. How can a private Cloud data store ensure passing critical data in pseudonymized form to the public Cloud?

*Forces*. It has to be ensured that all data classified as critical is pseudonymized in the data set passed to the public Cloud in order to prevent that the pseudonymization can be broken, e. g., by using tools for data analysis and data mining. The relation between the pseudonymized data and the critical data has to be stored in the private Cloud and prevented to be passed to the public Cloud. This will enable the relation of the pseudonymized processing results from the public business layer to the critical data.

*Solution*. A pseudonymizer of critical data is placed within the Cloud infrastructure of the private Cloud data storage. All requests to the private Cloud data storage have to be directed to the pseudonymizer. The private Cloud data storage is only reachable by the pseudonymizer of critical data. Results of requests for critical data originating off-premise are pseudonymized.

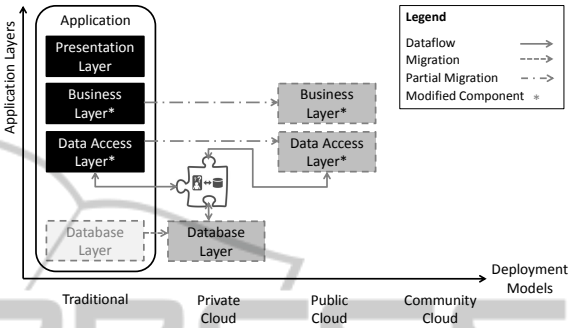



Figure 5: Sketch of Pseudonymizer of Critical Data.

*Sidebars*. The data contained in the private Cloud data store has to be categorized into critical and non-critical data, e. g., by applying the security classification described in the security regulations of the NATO and annotating the data accordingly. Both parts of the data access layer hosted traditionally and in the public Cloud have to be reconfigured to access the database layer only through the Pseudonymizer of Critical Data. We denote the required reconfigurations as "Data Access Layer*" in Figure 5.

The method for pseudonymization has to be configured and the relation has to be stored in the private Cloud data store.

*Results*. Assuming the data transfer from and to the private database layer happens exclusively through the Pseudonymizer of Critical Data, only non-critical data and pseudonymized data is passed to the public Cloud.

*Example*. A company started an initiative to improve there image with respect to Green computing. An external partner has been contracted to continuously monitor the relevant data and to report monthly on energy savings and energy saving potentials. As a result the external company gets access to the data. As the company does not want to reveal its internal secrets the data is pseudonymized.

*Next*. In case passing (and thus disclosing) pseudonymized critical data is not applicable, the Filter of Critical Data or the Anonymizer for Critical Data have to be considered.

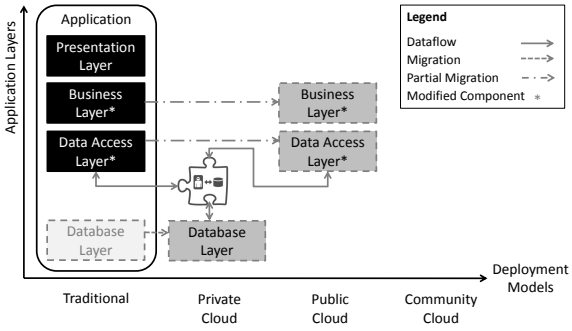



Figure 6: Sketch of Anonymizer of Critical Data.

## 2.5 Anonymizer of Critical Data

*Context*. The private Cloud data store contains critical and non-critical data. The business layer is partially moved to the public Cloud and needs access to data. To prevent disclosure and misuse the critical data is anonymized before being passed to the public Cloud. The logic implemented in the business layer is split into one requiring critical data and one, where critical data in anonymized form is sufficient for processing. This modification of the original business layer is denoted by "Business Layer*" in Figure 6. The party implementing or hosting the processing for which critical data in anonymized form is sufficient may not be trusted. It is not required to relate the anonymized processing results from the public business layer to the critical data.

*Challenge*. How can a private Cloud data store ensure passing critical data in anonymized form to the public Cloud?

*Forces*. It has to be ensured, that all data classified as critical is sufficiently anonymized in the data set passed to the public Cloud in order to prevent that the anonymization can be broken, e. g., by using tools for data analysis and data mining.

*Solution*. An Anonymizer of Critical Data is placed within the Cloud infrastructure of the private Cloud data store. All requests to the private Cloud data store have to be directed to the anonymizer. The private Cloud data store is only reachable by the Anonymizer of Critical Data. Results of requests for critical data originating off-premise are anonymized.

*Sidebars*. The data contained in the private Cloud data store has to be categorized into critical and non-critical data, e. g., by applying the security classification described in the security regulations of the NATO and annotating the data accordingly. Both parts of the data access layer hosted traditionally and in the public Cloud have to be reconfigured to access the database layer only through the Anonymizer of Critical Data. We denote the required reconfigurations as "Data Access Layer*" in Figure 6.

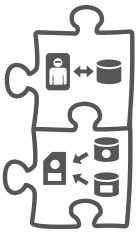
The method for anonymization and the respective granularity have to be configured. For instance, an off-premise application may retrieve anonymized data for each person and another may only use data aggregated for groups of 1 000 persons.

*Results*. Assuming the data transfer from and to the private database layer happens exclusively through the Anonymizer of Critical Data, only non-critical data and anonymized data is passed to the public Cloud.

*Example*. The database layer of an application for managing the customers of a health insurance built on Oracle Corporation MySQL, version 5.1 is moved to Amazon VPC choosing an AMI running MySQL 5.1 relational database on OpenSolaris. Thus, the database functionality remains the same. The database contains the personal information as well as the medical history of the customers. The business logic calculating general statistics is outsourced to the public Cloud. As there is no personal information of the customers required to calculate the general statistics such as how many people were suffering from a particular illness in the last year, only anonymized data is passed by the Anonymizer of Critical Data to the business layer in the public Cloud.

*Next*. In case passing (and thus disclosing) anonymized critical data is not applicable, the Filter of Critical Data has to be considered.

## 2.6 Composition of Cloud Data Patterns



Patterns of a pattern language are related to each other, they have to be considered as a whole and to be composable (Hohpe and Woolf, 2003). Thus, we have chosen the form of a piece of a puzzle for the pattern icons. We do not claim that all Cloud Data Patterns are composable with each other. Whether two or more Cloud Data Patterns are composable depends on the semantic and functionality of each of the patterns. Moreover, the specific requirements and context of the needed solution effect whether a composition of patterns is required. A composition of the Confidentiality Level Data Aggregator and the Anonymizer of Critical Data allows for the off-premise application part to access the private data stored in different data stores in anonymized form. The detailed investigation of compositions of Cloud Data Patterns is out of scope of this paper.

## 3 RELATED WORK

Pattern languages to define reusable solutions for recurring challenges have been first proposed in architecture by Christopher Alexander et al. (1977, 1979). Various publications on patterns exist in computer science that describe recommendations on how to face recurring challenges in different domains. In the following selection we focus on enterprise integration, application architecture, Cloud computing, data, security, and privacy in the order they are mentioned.

Hohpe and Woolf (2003) investigate problems developers have to solve when building and deploying messaging solutions for enterprise integration. We reuse the pattern form introduced by Hohpe and Woolf for describing the Cloud Data Patterns.

Fowler et al. (2002) provide forty reusable solutions for developing enterprise applications. In contrast to our work, database hosting in the Cloud is not considered. Similar to Fowler et al. our patterns are independent from the concrete Cloud technology used.

ARISTA Networks, Inc. (2009) provides seven patterns for Cloud computing. The only pattern dealing with data in the Cloud is the Cloud Storage Pattern. Fehling et al. (2011) provide architectural patterns to design, build, and manage applications using Cloud services. They focus on the general architecture. Solutions for moving or building the database layer in the Cloud are not provided. Nock (2008) provides 25 patterns for data access in enterprise applications. Nock does not treat Cloud data stores as we do.

Schumacher et al. (2006) present reusable solutions for securing applications and to integrate security design in the broader engineering process, but do not deal with data pseudonymization, data anonymization, and data filtering. Hafiz (2006) presents a privacy design pattern catalog. Hafiz achieves anonymity by mixing data with data from other sources instead of providing a general pseudonymization, anonymization, or filtering pattern.

The handling of personal data in the Cloud in Germany is regulated by §11 of the German Federal Data Protection Law (Federal Ministry of Justice, 1990). The law distinguishes between pseudonymization and anonymization, which we reuse in our confidentiality patterns. To the best of our knowledge, the patterns presented in this paper have not been outlined by other authors.

## 4 CONCLUSIONS

This work presented five reusable solutions to face challenges of avoiding disclosure of confidential data when moving the database layer to the Cloud or designing an application using a database layer in the Cloud. The list of the patterns has been harvested during our work with industry partners and research projects. We do not claim that the list of patterns is complete.

The presentation of the patterns did not go into technical details. For instance, scalability and single point of failure has not been treated. A possible scalability mechanism and a counter-measure is to implement each pattern using a hotpool in the Cloud. A hotpool consists of multiple instances of the component and a watchdog.

Our next step is a description of the general composition method and an evaluation using existing applications to be migrated to the Cloud.



## ACKNOWLEDGEMENTS

This work has been funded by the EU (4CaaSt, FP7/258862) and the BMWi (CloudCycle, 01MD11023). We thank Vasilios Andrikopoulos (IAAS), Gerd Breiter (IBM), and Olaf Zimmermann (IBM) for their valuable input.

## REFERENCES

Alexander, C. (1979). *The Timeless Way of Building*. Oxford University Press.

Alexander, C. et al. (1977). *A Pattern Language. Towns, Buildings, Construction*. Oxford University Press.

Amazon.com, Inc. (2011a). Amazon Relational Database Service.

Amazon.com, Inc. (2011b). Amazon Virtual Private Cloud.

ARISTA Networks, Inc. (2009). Cloud Networking: Design Patterns for Cloud-Centric Application Environments.

Armbrust, M. et al. (2009). Above the Clouds: A Berkeley View of Cloud Computing. Technical Report UCB/EECS-2009-28, EECS Department, University of California, Berkeley.

Department of Defense Security Institute (1993). International Programs Security Handbook, Chapter 10: NATO Security Procedures.

Dunkel, J. et al. (2008). *Systemarchitekturen fuer Verteilte Anwendungen*. Hanser Verlag.

Federal Ministry of Justice (1990). German Federal Data Protection Law.

Fehling, C. et al. (2011). An Architectural Pattern Language of Cloud-based Applications. PLoP'11. ACM.

Fowler, M. et al. (2002). *Patterns of Enterprise Application Architecture*. Addison-Wesley Professional.

Hafiz, M. (2006). A Collection of Privacy Design Patterns. PLoP'06.

Hohpe, G. and Woolf, B. (2003). *Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions*. Addison-Wesley.

Mell, P. and Grance, T. (2009). Cloud Computing Definition. *National Institute of Standards and Technology*.

Nock, C. (2008). *Data Access Patterns: Database Interactions in Object Oriented Applications*. Prentice Hall PTR.

Oracle Corporation (2011). MySQL.

Schumacher, M. et al. (2006). *Security Patterns: Integrating Security and Systems Engineering*. Wiley.

Unger, T. and Bauer, T. (2008). Towards a Standardized Task Management. MKWI'08. GITO-Verlag, Berlin.