Erschienen in: Springer Fachmedien Wiesbaden GmbH, ein Teil von Springer Nature 2018, S. Reinheimer (Hrsg.), *Cloud Computing*, Edition HMD; https://doi.org/10.1007/978-3-658-20967-4_12

Security-by-Design in der Cloud-Anwendungsentwicklung[1]

Gunther Schiefer, Andreas Oberweis, Murat Citak, Andreas Schoknecht
Karlsruher Institut für Technologie (KIT), Institut AIFB, Kaiserstr. 89, 76133 Karlsruhe
<vorname>.<nachname>@kit.edu, Tel.: +49 (721) 608 43509

Schlüsselwörter: Cloud Computing, Sicherheit, kontextbasiertes Zugriffsmodell, durchsuchbare Verschlüsselung, Security-by-Design

Abstract

Unternehmen erkennen zunehmend die ökonomischen und operationalen Vorteile von Cloud Computing, die es ihnen ermöglichen, sowohl signifikante Kosteneinsparungen zu erzielen als auch den Einsatz neuer Software-Anwendungen zu beschleunigen. Der Einsatz von Cloud Computing erfordert jedoch eine zunehmende Betrachtung neuer Herausforderungen an die Sicherheit von Daten, die immer noch eine Barriere für eine breitere Akzeptanz von Cloud Computing sind. In diesem Artikel werden Erkenntnisse aus dem von der EU geförderten Projekt PaaSword vorgestellt, welches das Ziel verfolgt, das Vertrauen in Cloud Computing zu erhöhen. In diesem Projekt wurde ein Datensicherheits-Framework entwickelt, wobei der Fokus auf Software-Entwicklern liegt, die bei der Entwicklung von sicheren Cloud-Anwendungen und –Diensten unterstützt werden sollen. Dazu wird zunächst das zugrundeliegende Architektur-Konzept vorgestellt, um dann auf die kontextbasierte Zugriffskomponente einzugehen. Zentraler Aspekt dieser Zugriffskomponente ist ein kontextbasiertes Zugriffsmodell, das von Entwicklern zur Annotation von Data Access Objects verwendet werden kann. Das Zugriffsmodell baut auf einem Attribute-based Access Control Modell auf. Dabei werden Zugriffsrechte gewährt, indem Zugriffsregeln ausgewertet werden, welche Kontextattribute berücksichtigen. Im PaaSword-Zugriffsmodell kann festlegt werden, auf welche Daten unter welchen Bedingungen zugegriffen werden darf. Die Formulierung der Regeln baut auf dem XACML-Standard auf, der es ermöglicht, einzelne Regeln mit Kontextbedingungen zu komplexeren Regelwerken zusammenzufassen. Weiterhin wird der Datenbankadapter für eine sichere Speicherung von Daten vorgestellt. Dieser agiert gegenüber einer Anwendung wie ein klassisches relationales Datenbanksystem, transformiert die Datenbank jedoch so, dass die Daten verschlüsselt gespeichert werden können und trotzdem durchsuchbar bleiben. Dazu werden mehrere Datenbanken und besonders gestaltete Indizes verwendet. Die sichere Speicherung wird unterstützt durch Maßnahmen zur sinnvollen Aufteilung von Attributen auf getrennte Datenbanken für die Indizes. Abschließend wird ein leichtgewichtiges Schlüsselmanagement beschrieben, welches durch eine Aufteilung des Datenbankschlüssels die Sicherheit weiter erhöht, eine weiteren Autorisierungsfaktor hinzufügt und die Mechanismen zur Zugriffskontrolle und Speicherung verbindet.

Abstract English

Companies increasingly recognize the economical and operational advantages of Cloud Computing enabling them to realize significant cost savings and to speed up the setup of software applications. Yet, the usage of Cloud Computing requires the consideration of new challenges regarding data security, which pose a serious threat to the adoption of Cloud Computing. This article presents results from the EU-funded PaaSword project aiming at increasing the trust in Cloud Computing. A data security framework has been developed during the project focusing on software developers, who shall be supported during the development of secure cloud applications and services. Therefore, firstly, the underlying architecture concept is introduced. The context-based access control component is described in further detail afterwards. The central aspect of this access control component is a context-based access control model, which can be used by developers to annotate data access objects. The access control model itself builds upon an attribute-based access control model. Thereby, access rights are granted through the evaluation of access rules taking context attributes into account. The PaaSword access control model determines under which circumstances an access request to which data is allowed. The formulation of such rules is based on the XACML standard, which allows combining single rules with context conditions to more complex policies. Furthermore, the database proxy for secure storage is introduced. This proxy acts like a normal relational database system but transforms the database in such a way that it is possible to store the data encrypted while still being able to search in it. Therefore, PaaSword uses multiple databases and special indexes. The secure storage is supported by measures for separating attributes according to privacy constraints. The article concludes with a short description of a lightweight key management. This breaks the database key into parts to increase privacy, to add another authorisation factor and to link access control and secure storage together.

[1] Vollständig überarbeiteter und erweiterter Beitrag basierend auf Schoknecht, Schiefer, Citak, Oberweis (2016) Security-by-Design in der Cloud-Anwendungsentwicklung, HMD – Praxis der Wirtschaftsinformatik Heft 311 53(5):688-697

Erschienen in: Springer Fachmedien Wiesbaden GmbH, ein Teil von Springer Nature 2018, S. Reinheimer (Hrsg.), *Cloud Computing*, Edition HMD; https://doi.org/10.1007/978-3-658-20967-4_12

1. Einleitung

Unternehmen erkennen zunehmend die ökonomischen und operationalen Vorteile von Cloud Computing, die es ihnen ermöglichen, sowohl signifikante Kosteneinsparungen zu erzielen als auch den Einsatz neuer Software-Anwendungen zu beschleunigen. Der Einsatz von Cloud Computing erfordert jedoch eine zunehmende Betrachtung neuer Herausforderungen an die Sicherheit von Daten, die laut dem LinkedIn Cloud Security Spotlight Bericht (Schulze 2015) die größte Barriere für eine schnellere Annahme von Cloud Computing sind. 90 % der 1.000 für diesen Bericht befragten Teilnehmer äußerten sich mäßig oder sehr besorgt bezüglich des Sicherheitsniveaus. In vielen Fällen werden jene Daten missbräuchlich verwendet, welche auf Datenträgern gespeichert sind. Im Arbeitsspeicher einer Cloud-Anwendung sind die Daten im Vergleich dazu in der Regel nur eine deutlich kürzere Zeitspanne verfügbar. Um den Schutz der Daten zu erhöhen, muss deshalb verstärkt der Schutz der persistent gespeicherten Daten ins Auge gefasst werden.

In diesem Artikel werden Erkenntnisse aus dem von der EU geförderten Projekt PaaSword[2] vorgestellt, welches das Ziel verfolgt, das Vertrauen in Cloud Computing zu erhöhen. Im Projekt wurde ein Datensicherheits-Framework entwickelt, wobei der Fokus auf Software-Entwicklern liegt, die bei der Entwicklung von sicheren Cloud-Anwendungen und –Diensten unterstützt werden sollen. Diese Entwickler können über Annotationen im Code beispielsweise bestimmen, welche Daten bei der Speicherung in einer Datenbank verschlüsselt werden sollen oder welche Kontextbedingungen bei einem kontextbasierten Zugriff gelten müssen. In dieser Hinsicht möchte das PaaSword-Projekt das Prinzip des Security-by-Design (Waidner et al. 2014) in der Praxis der Software-Entwicklung fester verankern. In diesem Artikel wird zunächst kurz das zugrundeliegende Architektur-Konzept vorgestellt, um dann vertieft auf die zwei zentralen Mechanismen für (i) die kontextbasierte Zugriffskontrolle und (ii) eine sichere Speicherung von Daten einzugehen.

Zentraler Aspekt der Zugriffskomponente ist ein kontextbasiertes Zugriffsmodell, das von Entwicklern zur Annotation von Datenzugriffsobjekten (sog. Data Access Objects, DAO) verwendet werden kann. Das Zugriffsmodell baut auf einem Attribute-based Access Control Model (ABAC) (Hu et al. 2014) auf. Bei ABAC werden Zugriffsrechte gewährt, indem Regeln ausgewertet werden, welche Kontextattribute berücksichtigen. Zu diesen Attributen können beispielsweise die IP-Adresse des anfragenden Computers, die Art des verwendeten Geräts, der momentane Aufenthaltsort oder die Rolle des Nutzers innerhalb einer Organisation gehören. Im PaaSword-Zugriffsmodell werden Aspekte konzeptualisiert, die bei der Auswahl von Datenzugriffsregeln beachtet werden müssen und mit deren Hilfe das kontextbasierte Zugriffsmodell festlegt, auf welche Daten unter welchen Bedingungen zugegriffen werden darf. Wenn die möglichen Kontextarten und deren Verarbeitung feststehen, können die Entwickler Zugriffsregeln zu Datenzugriffsobjekten hinzufügen. Im Projekt werden dazu die Möglichkeiten von Annotationen in Java genutzt und erweitert. Die Formulierung der Regeln baut auf dem XACML-Standard (Parducci et al. 2013) auf. XACML ermöglicht es, einzelne Regeln mit Kontextbedingungen anhand von Kombinationsalgorithmen zu komplexeren Regelwerken („Policies“) zusammenzufassen. Zudem wird eine ontologische Beschreibung von Regeln eingesetzt, die z. B. eine automatische Plausibilitätsprüfung ermöglicht.

Für die sichere Speicherung der Daten wurde in PaaSword der Datenbankadapter (Database Proxy) aus dem Projekt MimoSecco (Achenbach et al. 2011) in der Funktionalität erweitert. Der Datenbankadapter agiert gegenüber einer Anwendung wie eine relationale Datenbank, verwendet im Hintergrund jedoch mehrere Cloud-Datenbanken. Durch eine ausgeklügelte Datenbanktransformation werden die eigentlichen Daten vor der Speicherung komplett verschlüsselt und zusätzlich Indizes erzeugt, deren Nutzinformation ebenfalls verschlüsselt ist. Dadurch besteht weiterhin die Möglichkeit, die Daten zu durchsuchen, ohne vorher die verschlüsselten Daten wieder komplett aus der Datenbank auslesen zu müssen. Weiter hinzugekommen ist die Möglichkeit, Beschränkung zum Schutz der Vertraulichkeit zu definieren (Privacy Constraints) und die erzeugten Indizes entsprechend zu fragmentieren.

Im folgenden Kapitel wird zunächst das PaaSword-Framework zur Entwicklung sicherer Cloud-Anwendungen vorgestellt, um den Überblick herzustellen. Kapitel 3 beschreibt das kontextbasierten Zugriffsmodell, dabei wird insbesondere auf die Formulierung von Zugriffsregeln eingegangen. Die verteilte durchsuchbare Verschlüsselung wird in Kapitel 4 näher beleuchtet. Der Artikel endet mit einer Zusammenfassung in Kapitel 5.

[2] www.paasword.eu

Erschienen in: Springer Fachmedien Wiesbaden GmbH, ein Teil von Springer Nature 2018, S. Reinheimer (Hrsg.), *Cloud Computing*, Edition HMD; https://doi.org/10.1007/978-3-658-20967-4_12

2. Entwicklung sicherer Cloud-Anwendungen mit Hilfe des PaaSword-Frameworks

Ziel des PaaSword-Projektes war die Entwicklung eines Frameworks, um sichere Cloud-Anwendungen entwickeln zu können, um somit die Verbreitung dieser Anwendungen zu unterstützen. Dabei stehen nicht einzelne Sicherheitsmechanismen wie neuartige Verschlüsselungsalgorithmen im Vordergrund, sondern Anwendungsentwickler sollen bei der Entwicklung durch das Framework unterstützt werden. Abb. 1 zeigt zunächst die konzeptuelle Architektur des PaaSword-Frameworks, mit dessen Hilfe Anwendungsentwickler Cloud-Anwendungen entwickeln können, die auf eine physisch verteilte und vollständig verschlüsselte Datenbank zugreifen. Zu den wesentlichen Bestandteilen des Frameworks zählen (i) der Policy Enforcement Mechanism zur Beschreibung und Verwaltung von Zugriffsregeln sowie zur Annotation von Data Access Objects und (ii) eine physisch verteilte und verschlüsselte Datenbank. Auf dieser Basis kann dann eine Cloud-Anwendung (iii) mit Hilfe des PaaSword-Frameworks entwickelt und mit den Laufzeitkomponenten (PaaSword Container) ausgeführt werden.

Im Folgenden werden zunächst die generellen Zusammenhänge innerhalb des PaaSword-Frameworks beschrieben, um dann anschließend in Kapitel 3 vertieft auf das kontextbasierte Sicherheitsmodell (Context-aware Policy Access Model) einzugehen.

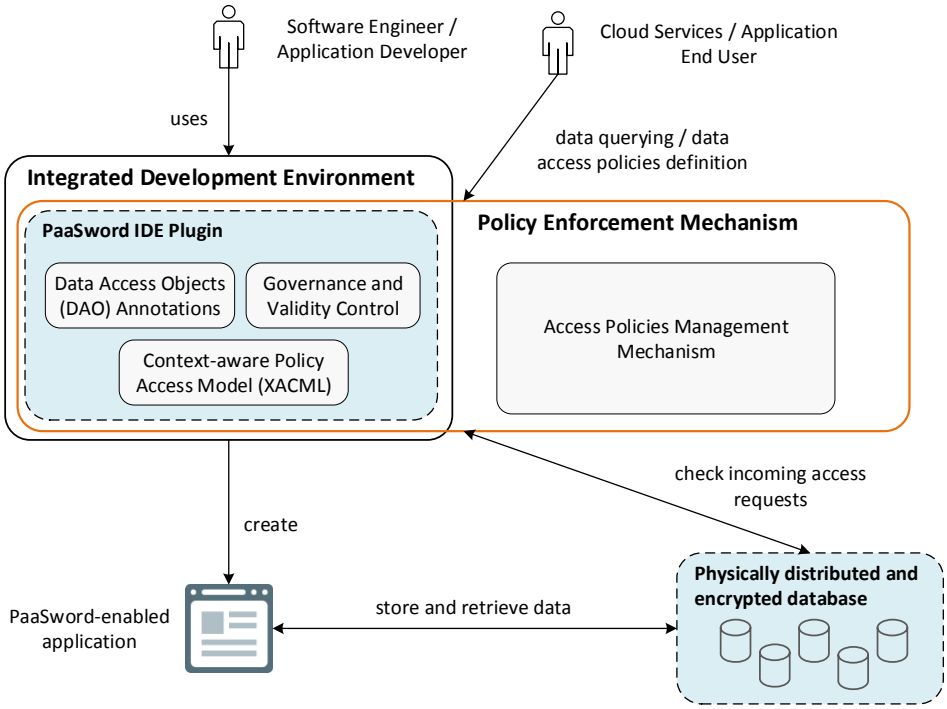

Abb. 1: Übersicht über die konzeptuelle PaaSword-Architektur

Im Rahmen des PaaSword-Projekts wurden Bibliotheken entwickelt, die bei der Entwicklung von Java-basierten Cloud-Anwendungen die Annotation von Data Access Objects ermöglichen. Diese Annotationen sind dazu gedacht, sensible Daten zu markieren, die geschützt werden sollen; um die Art der Verschlüsselung und Zugriffsrechte festzulegen oder um die Restriktionen bei der Verteilung der Attributwerte zu beschreiben. Bei der Kompilierung einer Anwendung werden solche DAO-Annotationen auf ihre Validität hin überprüft und mit dem restlichen Code kompiliert. Der Policy Enforcement Mechanism überprüft dabei die Validität der beschriebenen Zugriffsregeln zur Kompilierzeit (Governance and Validity Control Komponente) und ist ebenso dafür zuständig, Zugriffsberechtigungen eingehender Anfragen zur Laufzeit zu bestimmen. Darüber hinaus können zur Laufzeit weitere Zugriffsregeln definiert werden, sodass Regeln auch flexibel während des Betriebs einer Anwendung angepasst werden können. Die letztgenannten Optionen werden über die Access Policies Management Mechanism-Komponente realisiert. Wenn eine PaaSword-aktivierte Anwendung instanziiert wird, erzeugt der Datenbankadapter - entsprechend der Vorgaben von Operatoren und Annotationen im Code - ein passendes Set von benötigten verteilten Datenbanken und die dazugehörigen kryptografischen Schlüssel. Zur Laufzeit agiert der Datenbankadapter gegenüber der Anwendung wie eine monolithische relationale Datenbank, sichert die gespeicherten Daten jedoch durch die Datenbanktransformation ab (siehe Kapitel 4).

Weitere Details zur konzeptuellen Architektur des PaaSword-Frameworks können aus der PaaSword Referenzarchitektur (Gouvas und Michalas 2015) entnommen werden.

Erschienen in: Springer Fachmedien Wiesbaden GmbH, ein Teil von Springer Nature 2018, S. Reinheimer (Hrsg.), *Cloud Computing*, Edition HMD; https://doi.org/10.1007/978-3-658-20967-4_12

3. Kontextbasiertes Sicherheitsmodell

Für die Festlegung von Sicherheitsregelwerken zum Erstellzeitpunkt und zur Laufzeit benötigt der Policy Enforcement Mechanism ein Sicherheitsmodell, mit dem diese Regelwerke definiert werden können. Damit kann ein Entwickler schon beim Entwickeln von Anwendungen festlegen, welche sensiblen Daten einen besonderen Schutz benötigen. Mit Hilfe des Modells kann einerseits ein Sicherheitsprofil für die Speicherung dieser Daten an den Zugriffspunkt gebunden werden, andererseits können kontextabhängige Zugriffsregeln für den Zugriff auf diese Daten festgelegt werden.

3.1. PaaSword-Sicherheitsmodell

Für die Zusammenfassung der genannten Aufgaben in einem Modell berücksichtigt das PaaSword-Sicherheitsmodell die folgenden Aspekte:

- Sicherheitsrelevante Kontextelemente (Security Context Elements)
- Zugriffsberechtigungen (Permissions)
- Kontextmuster (Context Pattern)
- Verteilungs- und Verschlüsselungselemente (Data Distribution and Encryption Elements, DDE)

Die ersten drei Aspekte werden für die Formulierung von Zugriffsregeln benötigt. Damit können Data Access Objects annotiert werden. Hierbei werden nicht nur statische Zugriffsregeln definiert, sondern auch kontextabhängige Zugriffsregeln für eine an die Situation dynamisch angepasste Zugriffskontrolle festgelegt. Dazu werden die zur Auswertung relevanten Kontextelemente festgelegt, auf deren Werten die jeweils aktuellen Zugriffsentscheidungen entsprechend der formulierten Zugriffsregeln beruhen. Kontextmuster stehen direkt in Verbindung mit sicherheitsrelevanten Kontextelementen, welche wiederum mit Zugriffsberechtigungen verknüpft sind. Die Kontextmuster werden verwendet, um Zugriffsentscheidungen abhängig von der bisherigen Nutzung zu treffen. Hierdurch können beispielsweise unübliche Datenzugriffe herausgefiltert werden oder es können als Informationsbarriere - im Sinne einer „Chinese Wall“ (Brewer und Nash 1989) - Zugriffe auf konkurrierende Datenteile verhindert werden, welche nicht vermischt werden dürfen.

Die sicherheitsrelevanten Kontextelemente beispielsweise sind in fünf Top-Level-Konzepte eingeteilt (Verginadis et al. 2016):

- Location: Beinhaltet Informationen über den Ort (z. B. physikalischer Standort oder Netzwerk-Standort), an dem die Daten gespeichert werden oder von dem aus auf die Daten zugegriffen wird.
- DateTime: Hierbei werden zeitliche Aspekte wie beispielsweise konkrete Zeitpunkte oder -intervalle betrachtet, die einen Zugriff charakterisieren.
- Connectivity: Dieses Kontextelement enthält Informationen, die in Beziehung zur Konnektivität stehen, welche von Subjekten verwendet werden, um auf sensible Daten zuzugreifen. Die Art des Geräts oder die Art der Verbindung sind Beispiele hierfür.
- Object: Objekte beziehen sich auf jede Art von Artefakten (wie z. B. (nicht) relationale Daten, Dateien oder Software-Artefakte), die gemäß ihrem Sensibilitätsgrad geschützt werden sollen.
- Subject: Subjekte stellen Agenten (wie z. B. Organisation, Personen oder Gruppen) dar, die Zugriff auf bestimmte Artefakte anfordern.

Die Verteilungs- und Verschlüsselungselemente (DDE) legen die Sicherheitsmechanismen für die Speicherung der Daten fest. Hier kann die kryptografische Stärke einer Verschlüsselung ebenso definiert werden wie die geforderte Fragmentierung oder die Verteilung von Indextabellen auf physisch verschiedene Datenbanken.

Aus Abb. 2 wird der Zusammenhang der Aspekte im kontextbasierten Sicherheits-Modell von PaaSword ersichtlich. Weitere Details zum kontextbasierten Sicherheitsmodell finden sich in dem PaaSword-Projektbericht Context-aware Security Model (Verginadis et al. 2016). Im folgenden Abschnitt wird näher auf die Formulierung von Zugriffsregeln unter Einbeziehung von Kontextregeln eingegangen.

Erschienen in: Springer Fachmedien Wiesbaden GmbH, ein Teil von Springer Nature 2018, S. Reinheimer (Hrsg.), *Cloud Computing*, Edition HMD; https://doi.org/10.1007/978-3-658-20967-4_12

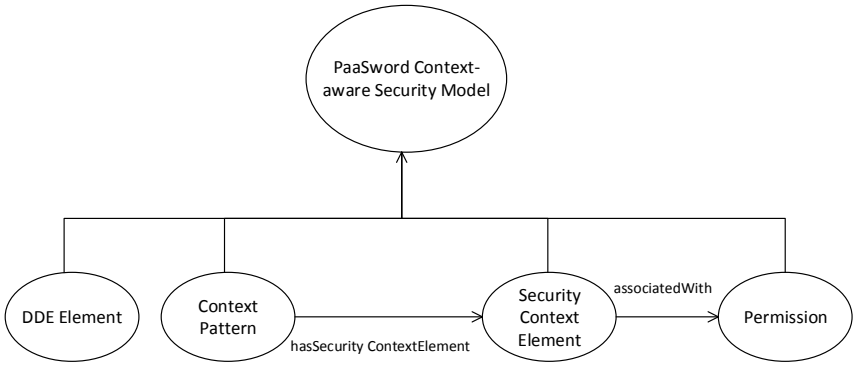

Abb. 2: Kontextbasiertes Sicherheitsmodell von PaaSword

3.2. Formulierung von Zugriffsregeln

Zugriffsmodelle sind dafür zuständig festzulegen, wer (welcher Nutzer) was (welche Operation) mit wem (welchen Objekten) durchführen darf. Objekte können dabei z. B. ein Dienst, ein Server, eine Datenbank oder eine einzelne Datenzeile in einer Datentabelle sein. Übliche Operationen sind schreiben, lesen, aktualisieren und löschen (CRUD[3]) von Daten. Der Nutzer ist das aktive Element und wird als Subjekt bezeichnet. Zugriffsregeln definieren damit die erlaubten Operationen, welche ein Subjekt mit einem Objekt durchführen darf.

Grundlegend können nach Decker (Decker 2011) folgende drei Formen der Zugriffskontrolle unterschieden werden, welche oft miteinander kombiniert werden: Discretionary Access Control (DAC), Mandatory Access Control (MAC) und Role-Based Access Control (RBAC). Beim Einsatz des DAC-Modells wird die Identität des Benutzers verwendet, um Zugriffsentscheidungen zu treffen. Die Zugriffe auf Objekte können durch Benutzer gesteuert werden, falls diese Eigentümer dieser Objekte sind, weshalb diese Form auch als benutzerbestimmbare Zugriffskontrolle bezeichnet wird. Bei MAC hingegen liegt die Kontrolle der Zugriffsberechtigungen nicht bei den Benutzern, sondern das System (letztlich der Sicherheitsbeauftragte) legt systemweite Zugriffsstrategien fest, die durch Benutzer nicht mehr beeinflussbar sind. Der zentrale Aspekt für die Kontrolle der Zugriffe bei RBAC ist die Verwendung von Rollen, um Objektzugriffe über die Rollenzugehörigkeit eines Benutzers zu steuern. Eine weitere bekannte Form der Zugriffskontrolle ist das Attribute-based Access Control Model (ABAC), welches die Formulierung von Zugriffsregeln basierend auf Attributen erlaubt. Grundsätzlich wurde gezeigt, dass sich die Zugriffsmodelle DAC, MAC und RBAC mit ABAC abbilden lassen (Priebe et al. 2005), sodass auch das PaaSword-Framework auf ABAC basiert.

Mit ABAC können beliebige Kontextattribute verwendet werden. Hier kann damit beispielsweise das gerade verwendete Endgerät, der momentane Aufenthaltsort oder die Art des verwendeten Zugangsnetzes (WLAN, Mobilfunk, usw.) als Kontextattribut genutzt werden. Ebenso kann die momentane Rolle eines Nutzers als dynamisches Attribut aufgefasst werden. Entsprechend dem ABAC-Modell werden Regelwerke (policies) als eine Sammlung von Regeln (rules) formuliert. Eine Regel legt fest, was ein Subjekt unter welchen Kontextbedingungen (context expression) für eine Autorisierung besitzt, um eine Aktion mit einem Objekt durchzuführen. In Abb. 3 ist eine ontologische Beschreibung der Zusammensetzung von Regelwerken dargestellt. Kontextbedingungen sind dabei eine beliebige Verknüpfung von Bedingungen der einzelnen Kontextelemente mittels der Booleschen Operatoren AND, OR, XOR und NOT. Diese Regewerke können nach ABAC noch zu Policy-Sets zusammengefasst werden, dies wird jedoch aus Gründen der Verständlichkeit im Weiteren weggelassen.

[3] **C**reate, **R**ead, **U**pdate, **D**elete

Erschienen in: Springer Fachmedien Wiesbaden GmbH, ein Teil von Springer Nature 2018, S. Reinheimer (Hrsg.), *Cloud Computing*, Edition HMD; https://doi.org/10.1007/978-3-658-20967-4_12

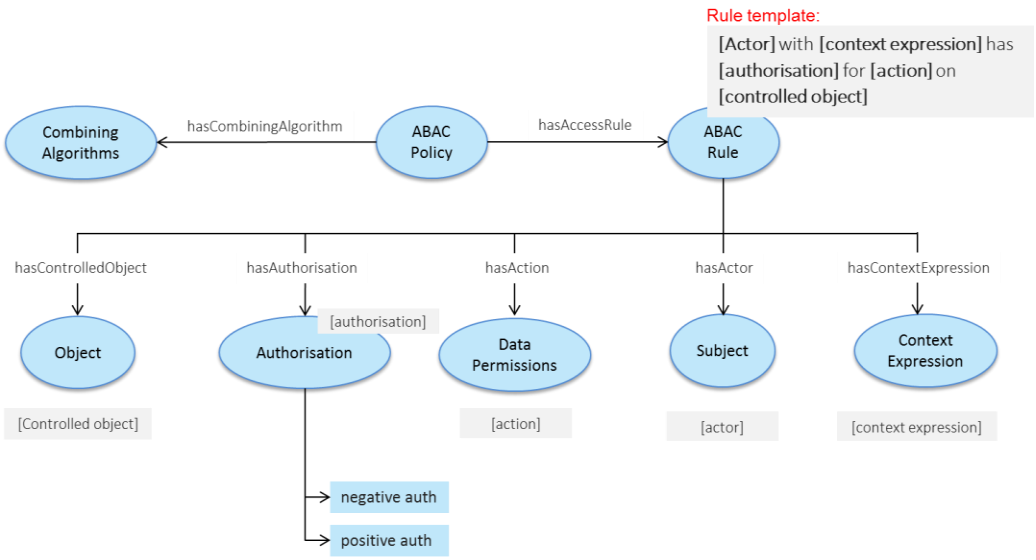

Abb. 3: Ontologische Beschreibung von Regelwerken (Verginadis et al. 2016)

Zudem bietet diese ontologische Repräsentation den Vorteil, die Kompatibilität von Zugriffsregeln durch automatisierte Reasoning-Verfahren überprüfen zu können. Sind z. B. mögliche, zulässige Örtlichkeiten für einen Datenzugriff bekannt, so können implementierte Regeln, die diese Ortsbeschränkungen überprüfen, mit diesem Wissen abgeglichen werden. Sollte ein unbekannter Ort in einer Regel spezifiziert sein, kann diese Regel abgelehnt werden (Veloudis und Paraskakis 2016). Dadurch können letztlich Entwickler bei der Beschreibung und Verwendung von Zugriffsregeln unterstützt werden, indem sie auf Inkonsistenzen oder Widersprüche aufmerksam gemacht werden.

3.3. Anwendung und Auswertung von Zugriffsregeln

Als Beispiel soll hier ein Parkhaus dienen, bei dem Kunden die Parkgebühren unter anderem mit ihrer Kreditkarte bezahlen können. Die Kreditkartendaten sind als schützenswert eingestuft und werden deshalb zusätzlich über Kontextbedingungen besonders geschützt. Für die Mitarbeiter im Parkhaus muss die Möglichkeit bestehen, aus den Daten zu entnehmen, ob ein Kunde die Parkgebühren bezahlt hat bzw. manuell Bezahlvorgänge vorzunehmen. Der Zugriff auf die Bezahldaten soll jedoch auf das Parkhaus und die Arbeitszeiten beschränkt werden. Durch diese Einschränkungen wird erreicht, dass die sensiblen Bezahldaten nicht unabsichtlich an öffentlichen Orten mit unbefugten Zuschauern von den Mitarbeitern eingesehen werden können, oder unbefugt von den Mitarbeitern von anderen Orten und außerhalb der Arbeitszeiten geändert werden. Dazu kann die folgende Regel festgelegt werden:

ParkingEmployee with (Location = CarPark AND Time = WorkingHour) has positive auth for Read on PaymentsTable

Diese durch das Template aus Abb. 3 beschriebene Regel realisiert einen Teil der oben erläuterten Zugriffskontrolle. Sie kontrolliert das Objekt Bezahldaten (*PaymentsTable*). Sie ist gültig für einen Nutzer (Subjekt), welcher die Rolle Parkhausmitarbeiter hat (*ParkingEmployee*) und für den gleichzeitig die Kontextbedingungen *Location=CarPark* und *Time=WorkingHour* zutreffen. Die Regel gibt eine Erlaubnis (*positive auth*) für das Lesen der Daten (*Read*). Um das oben beschriebene Szenario zu vervollständigen, muss eine zweite Regel definiert werden, bei der die erlaubte Aktion „*Write*“ anstelle von „*Read*“ lautet. Zusätzlich existiert eine Kombinationsregel für das Regelwerk, welche den Datenzugriff verbietet solange keine gültige Regel eine explizite Erlaubnis gibt. Bei der Auswertung der Regeln können die in Tabelle 1 dargestellten Fälle auftreten:

Subject	Object	Action	Context	Decision
Any person who is not a parking employee	`PaymentsTable`	`Read` or `Write`	Any context	Deny
`ParkingEmployee`	Any object other than `PaymentsTable`	`Read` or `Write`	Any context	Deny
`ParkingEmployee`	`PaymentsTable`	`Read` or `Write`	Location: Any other than `CarPark`	Deny
`ParkingEmployee`	`PaymentsTable`	`Read` or `Write`	Time: Any other than `WorkingHour`	Deny

Erschienen in: Springer Fachmedien Wiesbaden GmbH, ein Teil von Springer Nature 2018, S. Reinheimer (Hrsg.), *Cloud Computing*, Edition HMD; https://doi.org/10.1007/978-3-658-20967-4_12

`ParkingEmployee`	`PaymentsTable`	`Read` or `Write`	Location: `CarPark` Time:`WorkingHour`	Permit

Tabelle 1: Auswertungsfälle der Zugriffsregel

Die jeweilige Entscheidung wird aus folgenden Gründen getroffen: Da es keine Regel gibt, welche anderen Nutzern als den Mitarbeitern des Parkhauses einen Zugriff auf die Bezahldaten gibt, wird der Zugriff verweigert (Zeile 1). Da zudem keine Regel existiert, die den Mitarbeitern des Parkhauses einen Zugriff auf andere Objekte als die Bezahldaten gibt, wird der Zugriff ebenso verweigert (Zeile 2). Die Auswertung der Kontextbedingungen in den Zeilen 3 und 4 ergibt keine gültige Auswertung, sodass die Regel nicht zutrifft und der Zugriff verweigert wird. Erst wenn auch die Auswertung der Kontextbedingungen dazu führt, dass die formulierte Regel anwendbar ist, wird der Zugriff gewährt (Zeile 5).

Ein Entwickler kann dabei etwa für die Kontextbedingung `WorkingHour` die Zeit von 8:00 – 18:00 Uhr angeben. Sollte das Wissen über die Arbeitszeiten der Mitarbeiter des Parkhauses (z. B. zwischen 7:00 und 20:00 Uhr) für die Überprüfung der Regeln vorhanden sein, so kann der Entwickler auf die widersprüchliche Regel automatisiert aufmerksam gemacht werden. Somit wird eine mögliche Fehlerquelle reduziert.

4. Sichere Speicherung

Aktuelle Regelwerke (z. B. die Europäische Datenschutzgrundverordnung EU-DSGV) fordern technologische Schutzmaßnahmen entsprechend dem Stand der Technik. Die Verschlüsselung der gespeicherten Daten ist eine solche Maßnahme. Dazu bieten gängige Datenbanksysteme die Möglichkeit, die gespeicherten Nutzdaten verschlüsselt abzulegen (z. B. MariaDB[4]). Der Schutz der gespeicherten Daten ist aber nur solange gewährleistet, wie der entsprechende Schlüssel nicht missbräuchlich genutzt werden kann. Deshalb ist ein entsprechendes Schlüsselmanagement notwendig.

PaaSword bietet mit der durchsuchbaren verteilten Verschlüsselung die Möglichkeit, Daten auf einen höheren Schutzlevel im Vergleich zu einer monolithischen verschlüsselten Datenbank zu speichern. Der folgende Abschnitt beschreibt zunächst die Datenbanktransformation. Die darauf folgenden Abschnitte beschäftigen sich mit dem Mechanismus zur Beschränkung der gemeinsamen Speicherung von Attributen in Indizes (Privacy Constraints) und dem verteilten Schlüsselmanagement.

4.1. Durchsuchbare verteilte Verschlüsselung

In der Cloud gespeicherte Daten sollten zur Verringerung des Missbrauchspotenzials verschlüsselt gespeichert werden. Der Ansatz, die Daten an der Verwendungsstelle vor der Speicherung komplett zu verschlüsseln hat jedoch den Nachteil, dass diese nicht mehr durchsucht werden können. Werden also nur auszugsweise Daten benötigt, müssen dennoch alle Daten wieder geladen, diese entschlüsselt und dann der gewünschte Auszug erstellt werden. Um diesen Nachteil zu umgehen, wird die ursprüngliche Datenbank durch den PaaSword-Datenbankadapter transformiert (siehe Abb. 4).

[4] mariadb.com/kb/en/library/data-at-rest-encryption/

Erschienen in: Springer Fachmedien Wiesbaden GmbH, ein Teil von Springer Nature 2018, S. Reinheimer (Hrsg.), *Cloud Computing*, Edition HMD; https://doi.org/10.1007/978-3-658-20967-4_12

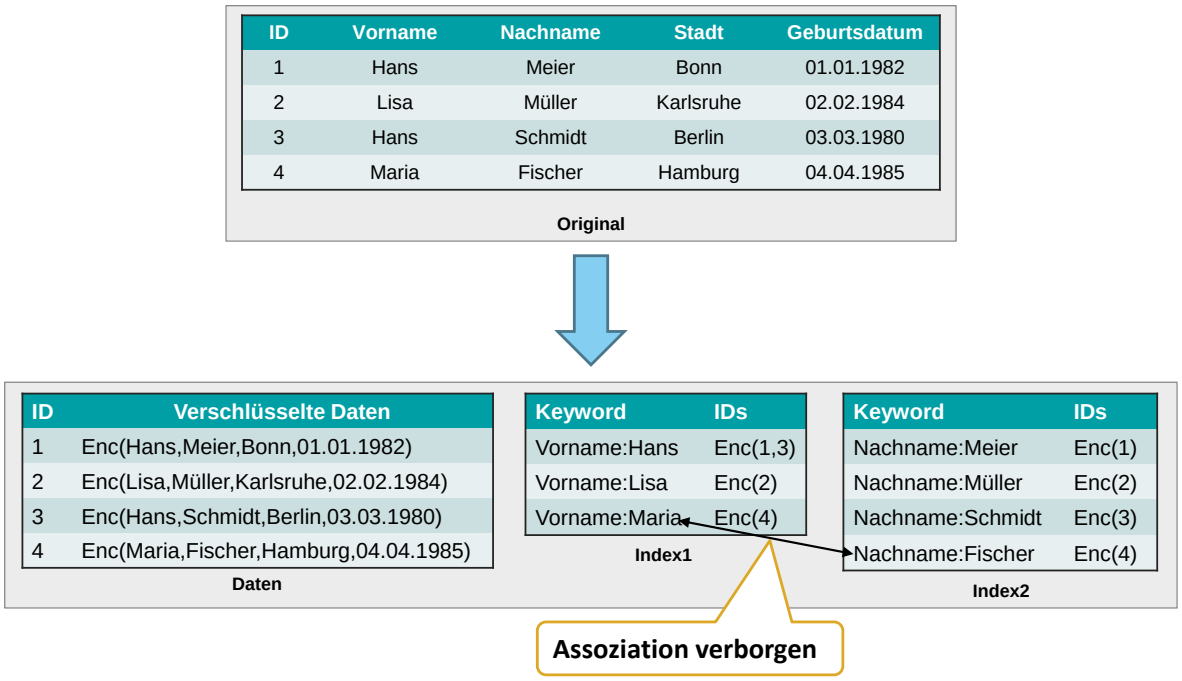

Abb. 4: Datenbanktransformation

Dazu verschlüsselt der Datenbankadapter die Originaldatentabelle zeilenweise und speichert diese auf einem Datenserver. Weiterhin werden Indextabellen der für eine spätere Suche benötigten Attribute erzeugt. Diese speichern zu den vorhandenen Attributwerten in verschlüsselter Form die jeweiligen Zeilen der verschlüsselten Originaltabelle, in welchen diese Attributwerte enthalten sind. Die Originaldaten sind damit komplett verschlüsselt und geschützt. Die einzelnen Indizes enthalten zwar noch alle Attributwerte, die Verbindung zwischen den Attributen ist jedoch verborgen. Die Datenbank könnte jetzt beispielsweise auch *Lisa Meier* oder *Hans Fischer* enthalten. Werden die Indizes auf verschiedene räumlich und organisatorisch getrennte Datenbankserver (Indexserver) verteilt, erhalten die Betreiber dieser Indexserver noch weniger Information über die Originaldaten.

Zum Auffinden der Daten - beispielsweise von *Hans Schmidt* - werden vom Datenbankadapter die Indizes entsprechend abgefragt. *Index1* liefert verschlüsselt (1,3) und *Index2* verschlüsselt (3) zurück. Der Datenbankadapter kann nun die Information der Indexserver entschlüsseln und die benötigte Datenzeile(n) ermitteln. Im nächsten Schritt werden diese Daten (und nur diese) aus der verschlüsselten Datentabelle abgerufen, entschlüsselt und der datenanfragenden Anwendung zur Verfügung gestellt.

Zur weiteren Erhöhung der Sicherheit können die Attributwerte verschlüsselt bzw. nur Hashwerte der Attributwerte in den Indextabellen abgelegt werden. Dadurch reduzieren sich allerdings auch die Abfragemöglichkeiten. Eine Suche nach Teilen von Attributwerten (z. B. SQL LIKE Befehl) ist dann nicht mehr möglich.

4.2. Verteilung der Attribute auf Indexserver

Wie oben schon angesprochen, kann die Verteilung der Indizes auf räumlich und organisatorisch getrennte Indexserver die Sicherheit weiter erhöhen. Praktisch dürfte es jedoch nicht sinnvoll sein, für jeden Index einen eigenen Servern vorzusehen. Es ist eher von einer begrenzten Anzahl an Indexservern auszugehen. In PaaSword können deshalb Regeln (Privacy Constraints) definiert werden, welche Attributwerte nicht zusammen abgelegt werden dürfen. Daraus wird dann eine Verteilung dieser Attributwerte für eine minimale Anzahl an benötigten Datenbankserver bestimmt.

Erschienen in: Springer Fachmedien Wiesbaden GmbH, ein Teil von Springer Nature 2018, S. Reinheimer (Hrsg.), *Cloud Computing*, Edition HMD; https://doi.org/10.1007/978-3-658-20967-4_12

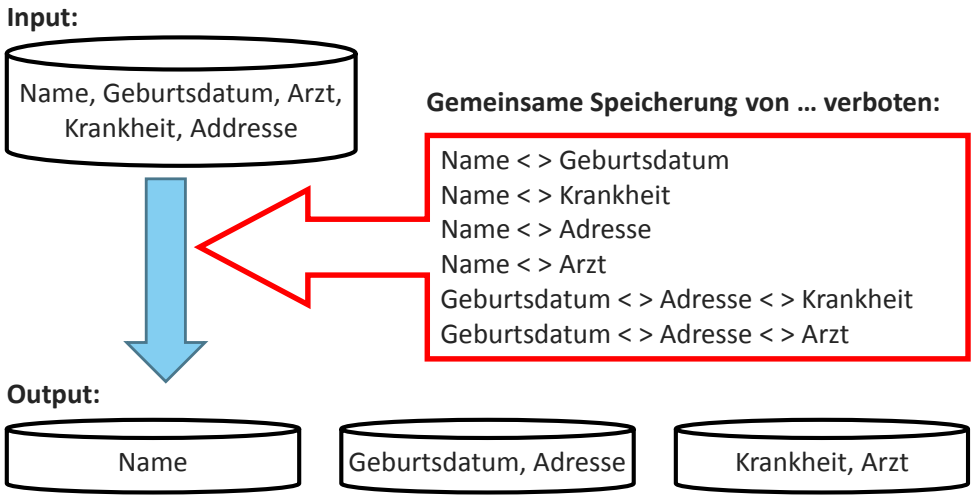

Abb. 5: Privacy Constraints

Für den Fall, dass die Verschlüsselung in einem Indexserver gebrochen und damit die Assoziation zwischen den Attributwerten dort wieder hergestellt werden könnte, soll der Informationsgehalt in einem Indexserver möglichst wenig Information offenbaren. Im Beispiel in Abb. 5 könnte beispielsweise die Zuordnung von Krankheit und Arzt in einem Indexserver offenbart werden. Der Informationsgewinn wäre jedoch sehr gering, da die Information, welcher Arzt welche Krankheiten behandelt, im Allgemeinen auch anderweitig öffentlich zugänglich ist.

4.3. Schlüsselmanagement

Daten sind durch eine Verschlüsselung nur dann geschützt, wenn der Schlüssel nicht bekannt ist. Die permanente Speicherung des Schlüssels direkt im Datenbankadapter eröffnet dem Betreiber einen jederzeitigen unkontrollierbaren Zugang zu den Daten. Der Schlüssel wird im Datenbankadapter jedoch nur während der Ver- und Entschlüsselung benötigt. Um das Risiko eines Missbrauchs oder eines Diebstahls zu reduzieren soll der Schlüssel dort nur dann vorliegen, wenn er benötigt wird. Ein Lösungsansatz ist z. B. ein dedizierter Schlüsselserver, der von einem vertrauenswürdigen Dritten betrieben wird und den Schlüssel nur auf eine autorisierte Anforderung hin herausgibt. Es kann auch sichere Hardware verwendet werden, welche den Schlüssel nur dann herausgibt, wenn eine Autorisierung des Dateneigentümers vorliegt (Schiefer 2012). Diese Autorisierung benötigt in der Regel eine eigene Infrastruktur mit Zertifikaten für alle Anwender. Die Autorisierung könnte auch darin bestehen, dass der eine Datenbankoperation anfragende Dateneigentümer den Schlüssel mitschicken muss. Wenn allerdings beispielsweise viele Mitarbeiter eines Unternehmens auf den gleichen Datenbestand mobil zugreifen müssen, wäre das Risiko für einen Schlüsselverlust recht hoch, da jeder Mitarbeiter im Besitz des Schlüssels sein müsste.

In PaaSword wurde ein leichtgewichtiger Ansatz gewählt. Die Grundidee ist, den Schlüssel mathematisch in drei Teile zu zerlegen und jeweils einen Teil beim Datenbankadapter, bei der Cloud-Anwendung und im Endgerät des Anwenders zu hinterlegen (siehe Abb. 6). Mit jeder Anfrage an die Cloud-Anwendung zur Durchführung einer Operation mit den Daten wird der Schlüsselteil des Anwenders gesichert mitgeschickt. Wenn die Cloud-Anwendung die Anfrage genehmigt - da der Anwender zu dieser Datenoperation berechtigt ist (siehe Zugriffskontrolle in Kapitel 3) – fügt sie ihren Schlüsselteil hinzu und veranlasst die nötigen Datenbankoperationen beim Datenbankadapter. Der Datenbankadapter fügt seinen dritten Schlüsselteil hinzu, rekonstruiert temporär den Schlüssel, kontrolliert dessen Validität anhand von Prüfdaten und führt im Erfolgsfall die geforderten Datenbankoperationen auf der transformierten Datenbank aus. Technische Details zum Verfahren finden sich u.a. in (Schiefer et al. 2017).

Erschienen in: Springer Fachmedien Wiesbaden GmbH, ein Teil von Springer Nature 2018, S. Reinheimer (Hrsg.), *Cloud Computing*, Edition HMD; https://doi.org/10.1007/978-3-658-20967-4_12

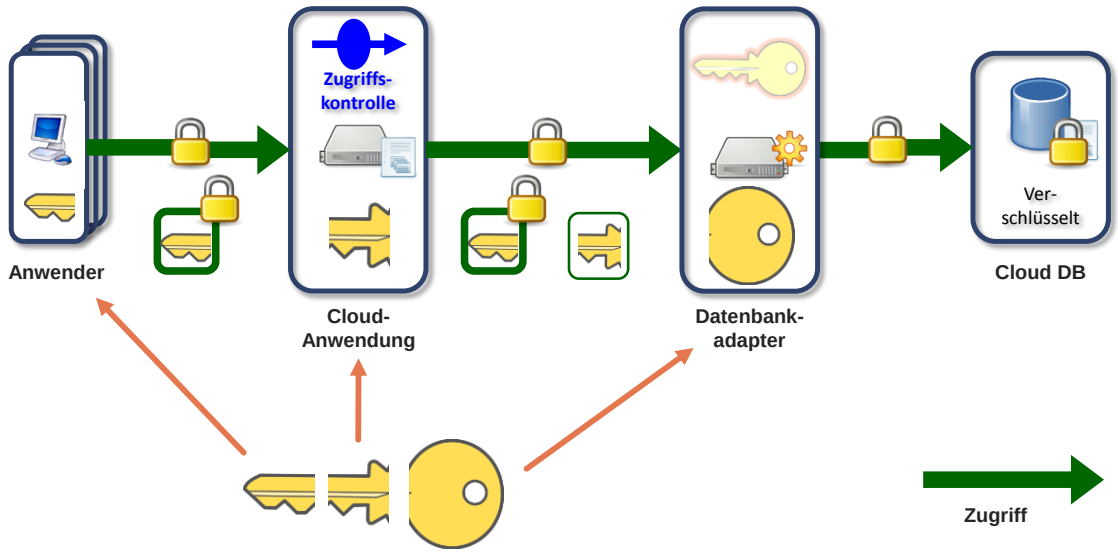

Abb. 6: Grundidee des Schlüsselmanagements

Mit diesem Verfahren werden mehrere Ziele erreicht: (i) Die Autorisierung durch den Anwender erfolgt ohne zusätzliche Infrastruktur. (ii) Es gibt keine Möglichkeit die Zugriffskotrolle zu umgehen (außer der Anwender wäre im Besitz von Schlüsselteil und privatem Zertifikat der Cloud-Anwendung). (iii) Der Schlüssel ist in keiner der beteiligten Komponenten als Ganzes gespeichert, er liegt lediglich kurzzeitig im Datenbankadapter vor.

Der Verlust eines Schlüsselteils beim Anwender ist ein möglicher Fall. Zur Sicherheit müssen dann umgehend die anderen Schlüsselteile gelöscht werden. Damit dies keine Auswirkung auf andere Anwender hat, werden für jeden Anwender individuelle Schlüsselteile erzeugt. Damit kann einem bestimmten Anwender problemlos der Zugang entzogen werden bzw. ein kompromittierter Schlüsselteil nutzlos gemacht werden, ohne dass dies Auswirkungen auf die anderen Anwender hat.

Für die Erzeugung der Schlüsselteile beim Aufsetzen des Systems bzw. beim Hinzufügen oder Ersetzen von anwenderspezifischen Schlüsselteilen ist der Originalschlüssel nötig. Dieser wird jedoch sicher beim Dateneigentümer (i.d.R. ein Anwenderunternehmen) verwahrt und nur kurzzeitig für diese Aufgabe „herausgeholt“. Ansonsten ist dieser in keinem laufenden System online verfügbar (kann also auch nicht entwendet werden) und ist keinem beteiligten Cloud-Anbieter bekannt. Die einzige Ausnahme ist die Zeit, in welcher er für die Durchführung einer Datenbankoperation im Datenbankadapter benötigt wird. Für den Fall der Fälle, dass z. B. der Betreiber der Cloud-Anwendung ausfällt, wäre das Anwenderunternehmen jedoch in der Lage, die eigenen Daten mithilfe des Originalschlüssels aus der verschlüsselten Datentabelle zu rekonstruieren. Damit stärkt das Verfahren die Datenhoheit des Dateneigentümers.

5. Zusammenfassung und Ausblick

Der Artikel gibt einen Einblick in die wesentlichen Sicherheitsmechanismen innerhalb des PaaSword-Projekts, welches Security-by-Design in der Cloud-Anwendungsentwicklung stärker verankern möchte. Es stellt dem Anwendungsentwickler die benötigten Werkzeuge zur Verfügung, um Sicherheitsregelwerke passend zu den jeweils gewünschten Sicherheitsanforderungen schon bei der Entwicklung umzusetzen. Dazu wurde dargelegt, wie mit Hilfe eines auf ABAC basierenden Modells Regeln zur Zugriffskontrolle beschrieben werden können. Die verwendete ontologische Repräsentation erlaubt zudem ein automatisiertes Schlussfolgern über Zugriffsregeln sowie weiterem Wissen, sodass die Kompatibilität von implementierten Regeln überprüft werden kann. Die durchsuchbare verschlüsselte Speicherung ist ein weiterer Baustein, mit dem die Sicherheit der persistenten Daten von Cloud-Anwendungen erhöht werden kann ohne auf die Vorteile der Durchsuchbarkeit zu verzichten. Das Schlüsselmanagement fügt einen weiteren Baustein hinzu, der die Umgehung der beiden andern Sicherheitsmechanismen unmöglich macht.

Damit verwendet PaaSword eine Drei-Faktor-Authentifizierung („Wissen“, „Sein“, „Haben“). Ein Anwender muss sich i.d.R. bei der Cloud-Anwendung anmelden (Faktor 1). Die kontextsensitive Zugriffskontrolle bietet die Option, Bedingungen über den Anwender festzulegen (z. B. Aufenthaltsort), welche er erfüllen muss (Faktor 2). Das Schlüsselmanagement bedingt, dass der Anwender im Besitz seines spezifischen Schüsselteils sein muss (Faktor 3). Die letzten beiden Faktoren werden außerdem nicht nur zu Beginn einer Sitzung abgefragt sondern mit jeder Aktion des Anwenders erneut geprüft.

Erschienen in: Springer Fachmedien Wiesbaden GmbH, ein Teil von Springer Nature 2018, S. Reinheimer (Hrsg.), *Cloud Computing*, Edition HMD; https://doi.org/10.1007/978-3-658-20967-4_12

Im Rahmen des Projektes wurden vier Pilotanwendungen und ein Mechanismus zur Unterstützung der Bereitstellung der benötigten Cloud-Ressourcen entwickelt. Diese demonstrieren eindrucksvoll die Möglichkeiten und das Potential für einen kommerziellen Einsatz der entwickelten Lösung, welcher durch die beteiligten Industriepartner erfolgt.

Acknowledgement: This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 644814.

6. Literatur

Achenbach D, Gabel M, Huber M (2011): MimoSecco: A middleware for secure cloud storage. In Frey D D, Fukuda S, Rock G, eds.: Improving Complex Systems Today, Proceedings of the 18th ISPE International Conference on Concurrent Engineering, July 4-8, 2011, Boston, MA, USA, Springer S. 175–181

Brewer D, Nash M (1989): The Chinese Wall Security Policy. In: Proceedings of IEEE Symposium on Security and Privacy. S. 206–214.

Decker M (2011): Modellierung ortsabhängiger Zugriffskontrolle für mobile Geschäftsprozesse. KIT Scientific Publishing, Karlsruhe

Gouvas P, Michalas A (2015): PaaSword Reference Architecture - Deliverable D1.3. https://www.paasword.eu/deliverables. Gesehen 01.12.2017

Hu V, Ferraiolo D, Kuhn R, Schnitzer A, Sandlin K, Miller R, Scarfone K (2014): Guide to Attribute Based Access Control (ABAC) Definition and Considerations. NIST Special Publication 800-162. http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-162.pdf. Gesehen 01.12.2017

Parducci B, Lockhart H, Rissanen E (2013): OASIS eXtensible Access Control Markup Language (XACML) TC http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-os-en.pdf. Gesehen 01.12.2017

Priebe T, Dobmeier W, Muschall B, Pernul G (2005): ABAC - Ein Referenzmodell für attributbasierte Zugriffskontrolle. Sicherheit 2005: Sicherheit - Schutz und Zuverlässigkeit, S. 285-296.

Schiefer G (2012): Wer liest alle meine Daten in der Wolke. OBJEKTspectrum Themenspecial Cloud Computing, SIGS DATACOM, Troisdorf

Schiefer G, Citak M, Schoknecht A, Gabel M, Mechler J (2017): Security in a distributed key management approach. In: IEEE 30th International Symposium on Computer-Based Medical Systems (CBMS), S. 816-821

Schulze H (2015): LinkedIn Cloud Security spotlight report. http://media.scmagazine.com/documents/114/cloud-security-spotlight-repor_28381.pdf. Gesehen 01.12.2017

Veloudis S, Paraskakis I (2016): Access Policies Model - Deliverable D2.2. https://www.paasword.eu/deliverables. Gesehen 01.12.2017

Verginadis Y, Patiniotakis I, Mentzas G (2016): Context-aware Security Model - Deliverable D2.1. https://www.paasword.eu/deliverables. Gesehen 01.12.2017

Waidner M, Backes M, Müller-Quade J (2014): Development of Secure Software with Security by Design. Fraunhofer Institute for Secure Information Technology, SIT Technical Reports, SIT-TR-2014-03.