B. Mitschang et al. (Hrsg.): BTW 2017 – Workshopband,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2017 237

Privatsphäre-schützende Bereichsanfragen in unsicheren Cloud-Datenbanken

Daniel Janusz[1] Jochen Taeschner[2]

Abstract: Online-Dienste benutzen zunehmend cloud-basierte Datenbanken. Wenn dabei sensible personenbezogene Informationen gespeichert werden, müssen die Daten vor unberechtigten Zugriffen geschützt werden. Dabei dürfen nur berechtigte Personen für vorher definierte Zwecke Daten abfragen. Durch Verschlüsselung kann der Zugriff eingeschränkt werden. Ferner lassen sich mittels Datenschutzpräferenzen für jedes Individuum in der Datenbank Zugriffszwecke speichern und validieren. Eine Bereichsanfrage auf verschlüsselte Attributwerte unter Beachtung aller individuellen Präferenzen wird bisher von keinem Datenbanksystem zufriedenstellen bearbeitet. In unserem Ansatz verzichten wir auf oft benutzte „Tricks“, wie die Verwendung von deterministischen Verschlüsselungen, die Angriffe auf die Kryptographie zulassen. Stattdessen benutzen wir zwei getrennte Komponenten für die beiden Aufgaben: einen verschlüsselten Index und eine Anfragebearbeitung, die tupelweise die individuellen Präferenzen auswertet.

Keywords: Range Queries, Encrypted Cloud Database, Privacy Policies.

1 Einleitung

Bei der Administration eines Datenbankmanagementsystems können viele Fehler gemacht werden. Das richtige Backup-Management oder das Einspielen von Softwareaktualisierungen stellen essentielle Aufgaben dabei dar. Die Expertise für alle anfallenden Aufgaben besitzen vor allem kleine Unternehme in der Regel nicht. Deshalb lagern viele Firmen ihr Datenmanagement zunehmend in cloud-basierte Datenbanksystem aus, welche sich um alle anfallenden administrativen Aufgaben kümmern.

Wenn die Speicherung von sensiblen personenbezogene Informationen an einen Cloud-Anbieter ausgelagert wird, sollten die Daten vor unberechtigten Zugriffen besonders geschützt werden. Da die Daten in einer potentiell angreifbaren Umgebung gespeichert werden, muss sichergestellt sein, dass nur berechtigte Personen für vorher definierte Zwecke Daten abfragen dürfen. Durch Verschlüsselung kann der Zugriff eingeschränkt werden, sodass nicht einmal der Cloud-Anbieter die Daten im Klartext lesen kann. Ferner lassen sich mittels Datenschutzpräferenzen für jedes Individuum in der Datenbank Zugriffszwecke speichern und validieren. Es gibt bereits viele Ansätze, die Lösungen für dieses Szenario vorschlagen, z.B. [Ne15] und [BL08].

Eine Bereichsanfrage auf verschlüsselte Attributwerte unter Beachtung aller individuellen Datenschutzpräferenzen wird bisher von keinem Datenbanksystem zufriedenstellen bearbeitet. In unserem Ansatz verzichten wir auf oft benutzte „Tricks“, wie die Verwendung von

[1] Humboldt-Universität zu Berlin, Institut für Informatik, janusz@informatik.hu-berlin.de
[2] Humboldt-Universität zu Berlin, Institut für Informatik, taeschnj@informatik.hu-berlin.de

homomorpher oder deterministischer Verschlüsselung, die ineffizient ist bzw. Angriffe auf die Kryptographie zulässt. Stattdessen benutzen wir zwei getrennte Komponenten für die beiden Aufgaben: einen verschlüsselten Index und eine Anfragebearbeitung, die tupelweise die individuellen Präferenzen für ein angefragtes Attribut auswertet. Der verschlüsselte Index basiert auf einem klassischen B-Baum, der effiziente Algorithmen zur Berechnung des Anfrageergebnisses benutzt. Die Daten bleiben hierbei jederzeit verschlüsselt. Vor der Rückgabe des Ergebnisses muss tupelweise die Einhaltung der Präferenzen sichergestellt werden, die von den Individuen für die Verwendung ihrer Daten angegeben wurden.

Das in dieser Arbeit vorgestellte Anfragebearbeitungssystem wurde prototypisch implementiert. Es bettet sich in eine größere Plattform zur sicheren und datenschutzkonformen Benutzerverwaltung als Online-Service [JTD16] ein. Bei der Implementierung des letzten Teils der Plattform zeigte sich, dass es keine zufriedenstellende Lösung für Bereichsanfragen auf ein einzelnes Attribut in dem angestrebte Szenario gibt. Den dabei entwickelten Ansatz stellt dieser Beitrag vor. Der Überblicksbeitrag aus dem Datenbankspektrum 02/2016 [JTD16] enthält als Lösung bisher noch einen Ansatz, der auf einem Kryptoprozessor basiert, was das Gesamtsystem jedoch teurer und plattformabhängig macht.

Im nachfolgenden Abschnitt 2 werden wir das genaue Anfrageszenario beschreiben und die dabei auftretenden Herausforderungen verdeutlichen. Ein Literaturüberblick in Abschnitt 3 wird erörtern, warum sich bisherige Ansätze nicht für das Szenario eignen. Danach präsentieren wir in den Abschnitten 4 und 5 unseren Lösungsansatz für die beiden Hauptherausforderungen. Der vorletzte Abschnitt 6 stellt das Gesamtsystem vor, wobei insbesondere auf das Zusammenspiel der zuvor vorgeschlagenen Einzelkomponenten eingegangen wird. Der Beitrag endet mit einer Zusammenfassung und einem Ausblick auf offene Fragestellungen.

2 Anfrageszenario

Wenn Nutzer eines Online-Dienstes sich neu registrieren, geben Sie in der Regel personenbezogene Daten wie z.B. Name oder E-Mailadresse an. Wird der Onlinedienst in einem Cloud-System betrieben, wie es z.B. von Amazon AWS angeboten wird, befinden sich die Nutzerdaten anschließend in der dort laufenden Datenbank. An dieser Stelle hat nicht nur der Service-Anbieter Zugriff auf die Daten, sondern auch der Cloud-Anbieter. Den Nutzern eines Online-Dienstes ist selten ersichtlich, wo seine Daten letztlich gespeichert werden. An dieser Stelle wäre es wünschenswert, dass die Daten vor der Speicherung so verschlüsselt werden, dass der Cloud-Anbieter die Daten nicht mehr im Klartext lesen kann. Das Passwort zum Entschlüsseln der Daten sollte nur durch den Onlinedienst benutzt werden. Dadurch bleiben die Daten selbst bei potentiellen unberechtigten Zugriffen durch Hacker oder auch Administratoren vor einer Kompromittierung geschützt.

Über die Verwendung und Weitergabe von personenbezogenen Daten erteilen Online-Dienste in ihrer Datenschutzerklärung Auskunft. Idealerweise sollten Benutzer jeden Verwendungszweck und jeder Weitergabe eines Datums feingranular zustimmen oder ablehnen können. Diese individuellen Datenschutzpräferenzen können zusammen mit den Daten

in Form von *Sticky Policies* [PM11] gespeichert werden. Somit kann bei jedem Zugriff auf ein Datum die Einhaltung der Präferenzen für die geplante Verwendung überprüft werden. Ganz allgemein sollten personenbezogene Daten nach den Prinzipien der *Hippocratic Databases* [Ag02] verwaltet werden. Einen Überblick über daraus entstandene konkrete Anforderungen und Ansätze findet sich in [BBL05]. Im Datenbankspektrum 02/2016 [JTD16] stellen wir eine Nutzerdatenverwaltung vor, welche versucht diesen Anforderungen gerecht zu werden, indem z.B. Nutzerdaten verschlüsselt und zusammen mit Nutzer-Präferenzen speichert werden. Für einen Nutzer, der einen Online-Dienst mit dieser Nutzerdatenverwaltung verwenden möchte, können seine verschlüsselten Daten über eine anonyme Tupel-ID aus der Datenbank abgefragt werden. Die Tupel-ID für einen Nutzer bekommt der Online-Dienst bei die Anmeldung des Nutzers übermittelt.

Der Online-Dienst kann in der Nutzerdatenverwaltung unter Angabe eines Zweckes einfache Punktanfragen auf einzelne verschlüsselte Tupel stellen oder alternativ die komplette verschlüsselte Tabelle abfragen. Hierbei wird immer sichergestellt, dass für jedes angefragte Tupelattribut eine Präferenzenauswertung bzgl. des angegebenen Zweckes stattfindet. Durch die Angabe eines Zwecks in einer Anfrage kann der Online-Dienst sich selbst absichern, dass die in der Antwort enthaltenen Nutzerdaten für den beabsichtigten Verwendungszweck wirklich verwendet werden dürfen. Es geht hierbei nicht darum den Online-Dienst am Datenzugriff allgemein zu hindern – diese Erlaubnis hat er vom Nutzer bereits vor der Datenspeicherung bekommen –, sondern es wird lediglich eine zweckgebundene Verwendung und Weitergabe der Nutzerdaten sichergestellt.

Für viele Prozesse eines Onlinedienstes genügen diese Anfragemöglichkeiten bereits, z.B. eine Bestellung durchführen. Sollen die Daten jedoch effizient ausgewertet werden, z.B. Analysen über einen Teil der Nutzerbasis, werden weitere Anfragetechniken benötigt. Insbesondere Bereichtsanfragen auf Attributwerte werden dabei benötigt. Da die Werte in den Tupel-Attributen jedoch verschlüsselt sind, werden dafür besondere Verfahren benötigt.

3 Private Database Outsourcing

Es gibt viele Ansätze, die den Zugriff auf sensible Daten verhindern und es trotzdem erlauben eine Datenbank in ein Cloud-System auszulagern. In dieser Abschnitt werden Techniken diskutiert, die Daten innerhalb einer Datenbank von unberechtigten Zugriff schützen können und dennoch effiziente Bereichsanfragen ermöglichen.

Die existierenden kryptographischen Verfahren bieten unterschiedliche Garantien. Sie lassen sich in Kategorien einteilen:

- **Homomorphe Verschlüsselung**: Diese Verfahren erlauben es Funktionen auf den verschlüsselten Daten zu berechnen. Bisher existiert allerdings keine *vollhomomorphe Verschlüsselung*, mit der beliebige Funktionen berechenbar sind.
- **Durchsuchbare Verschlüsselung**: Durch diese Verfahren können verschlüsselte Dokumente nach einem Auftreten vor Schlüsselwörtern durchsucht werden.

- **Geordnete Verschlüsselung**: Eine Menge von verschlüsselten Werten besitzt bei diesen Verfahren eine Reihenfolge mit der Eigenschaft: Wenn $x \leq y$, dann gilt $Enc(x) \leq Enc(y)$.
- **Deterministische Verschlüsselung**: Hierbei wird ein Wert bei der Verwendung eines festen Schlüssels immer in den gleichen Ciphertext überführt.
- **Probabilistische Verschlüsselung**: Hierbei wird ein Wert bei der Verwendung eines festen Schlüssels immer in den neuen Ciphertext überführt.

Die Verschlüsselungsverfahren aus den genannten Kategorien werden in diverse Arbeiten in der Datenbankliteratur benutzt. TrustedDB [BS11] und Cipherbase [Ar13] erlauben SQL-Anfragen an die probabilistisch-verschlüsselte Datenbank. Sie bieten den vollen Umfang von Anfragemöglichkeiten einer typischen SQL-Datenbank an. Beide Systeme greifen dabei auf einen Kryptoprozessor (engl. secure co-processor) zurück, der die verschlüsselten Daten in der Cloud auswertet. Dieses Verfahren hat mindesten zwei entscheidende Nachteile: Einerseits müssen alle Nutzer darauf vertrauen, dass der Kryptoprozessor bei der Verarbeitung ihre sensiblen Daten keine Fehler macht und nicht geknackt werden kann. Andererseits lässt sich schwer abschätzen wie gut die Benutzung solcher Prozessoren bei einer steigenden Nutzeranzahl und Datenmenge skaliert. Dies dürfte außerdem auch die Kosten der Datenhaltung erheblich steigern. CryptDB [Po11] verzichtet auf zusätzliche Hardware und verwendet alternativ ein zwiebelartiges Verschlüsselungssystem. Die Entwickler setzen auf eine Reihe unterschiedlicher Verschlüsselungsverfahren unter Ausnutzung ihrer Stärken und Schwächen. Die äußeren Hüllen der Zwiebel beruhen auf besonders sicheren kryptographischen Verfahren. Während die inneren Schichten bestimmte Rückschlüsse und damit Auswertungsmöglichkeiten zulassen, z.B. eine Ordnung auf den Daten. Die Zwiebelhüllen werden jeweils nur soweit entschlüsselt, wie es für die Anfrage notwendig ist. Durch diese Entschlüsselung in der Cloud werden die Daten jedoch angreifbar.

Eine weitere Klasse von Anfragetechniken versucht völlig ohne eine Entschlüsselung der Daten auszukommen. Bisher gibt es allerdings kein Verfahren, das die vollständige SQL-Anfragevielfalt auf probabilistisch-verschlüsselten Daten bietet. Effiziente Algorithmen existieren nur für Teilbereiche. Zuerst entstanden Forschungsansätze für eine Schlüsselwortsuche [Sh05]. Hierzu gibt es effiziente Verfahren, die eine genaue Übereinstimmung des gesuchten Schlüsselwortes mit einem verschlüsselten Attributwert identifizieren können [YZW06]. Um jedoch Bereichsanfragen ausführen zu können, müssen alle bisher existierenden Verfahren einige „Tricks“ anwenden. Wie bei CryptDB werden die Daten mit nicht-probabilistischen kryptographischen Verfahren verschlüsselt, die Rückschlüsse auf die Daten zulassen. Die meisten dieser Verfahren sind effizient, lassen sich aber mittels statistischer Analyse oder der Auswertung von Zugriffsmustern angreifen. Alternativ kann ein Angriff vermieden werden, wenn dafür die Effizienz verringert wird. Beispielsweise kann die von Rivest et al. [RAD78] vorgeschlagene homomorphe Verschlüsselung Berechnung eines Schaltkreises durchführen und damit jedes Computerprogramm nachbilden. Jedoch sind die Implementierungen ([SWP00] [Ge09]) bisher extrem rechenaufwendig und nicht praktikabel.

In der Regel benutzen Datenbanken für die effiziente Berechnung von Bereichsanfragen einen Index. Index-basierte Anfragebearbeitung wurden für probabilistisch-verschlüsselte Daten adaptiert [Sh05]. Durch *Private Indexing* [Go03] kann ein nicht-vertrauenswürdiger Server zur Berechnung einer Bereichsanfrage benutzt werden. Wang et al. [WAE11] präsentieren in ihrer Arbeit einen sicheren B+-Baum für eine effiziente Bearbeitung von Datenbankanfragen. Verfahren, die auf verschlüsselten Indizes basieren, benötigen keine Spezialhardware oder vertrauenswürdige Dritte und bieten somit einen geeigneten Ansatzpunkt für den in dieser Arbeit beabsichtigten Einsatzbereich. Jedoch kann der Index nur auf Basis der unverschlüsselten Daten erstellt werden. Mittels durchsuchbarer Verschlüsselung können Indizes entworfen werden, die auch verschlüsselte Werte in den Index einfügen können. Neuhaus et al. [Ne15] haben die für einen Einsatz in Cloud-Systemen optimierte Datenbank MongoDB mit einem solchen Index für eine effiziente Stichwortsuche erweitert. Obwohl alle genannten Verfahren die Daten probabilistisch verschlüsseln, werden jedoch die Metadaten in Form der entworfenen Indizes nicht probabilistisch verschlüsselt. Es bleibt also auch hier ein potentieller Angriffsvektor übrig, durch den die Nutzer-Daten kompromittiert werden könnten. Sollen alle Daten probabilistisch verschlüsselt werden, kann keines der existierenden Anfrageverfahren benutzt werden.

4 Bereichsanfragen auf verschlüsselte Daten

Es gibt kein effizientes Verfahren, um Bereichsanfragen auf probabilistisch verschlüsselte Daten innerhalb einer Clouddatenbank zu bearbeiten. Allerdings kann durch eine alternative Anfragebearbeitung das Anfrageszenario mit cloud-seitig gespeicherten verschlüsselten Daten dennoch ausgeführt werden. Die vorgestellten index-basierten Verfahren wären effizient und sicher, wenn auch der Index probabilistisch verschlüsselte wäre. Jedoch kann dadurch ein Cloud-Server den Index nicht mehr auswerten. Im Falle eines B-Baumes kann insbesondere die Entscheidung, in welchem Kindknoten sich der Suchschlüssel befindet, nicht mehr getroffen werden. Wenn genau diese Entscheidung an den Anfragenden delegiert würde, wäre das Problem gelöst und die Bereichsanfrage könnte beantwortet werden. Der Anfragenden muss den Schlüssel besitzen, da er sonst die Antwort nicht entschlüsseln könnte. Also kann er auch die an ihn delegierte Entscheidung berechnen. Dabei kann der Index nicht kompromittiert werden, weil der Anfragende grundsätzlich die verschlüsselten Daten zugreifen darf. *ZeroDB* [EW16] implementiert genau den skizzierten Ansatz. In Abbildung 2 wird der Suchablauf beispielhaft für den in Abbildung 1 dargestellten B-Baum für den Wert „11“ illustriert.

Durch die Auslagerung von Teilschritten aus der server-seitigen Datenbank an den anfragenden Client wird der Angriffsvektor auf den Index beseitigt. Dies geschieht auf Kosten der Anfragekomplexität, die um teure Kommunikationsschritte erweitert werden muss. Da B-Bäume in der Regel aber keine große Höhe besitzen, sind in der Praxis nur wenige Kommunikationsrunden (Anzahl = Höhe) notwendig. Die sich daraus ergebene erhöhte Latenz der Anfrage bleibt selbst bei einem B-Baum mit 1 Millionen Einträgen unter einer Sekunde [EW16]. Für einen Online-Dienst bleibt die Antwortzeit auf eine Anfrage damit in einem akzeptablen Rahmen.

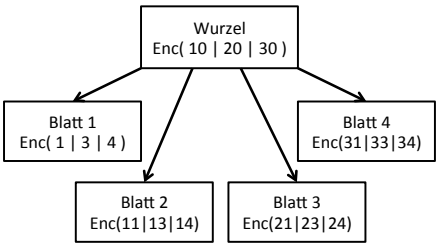

Abb. 1: Beispiel für einen verschlüsselten B-Baum

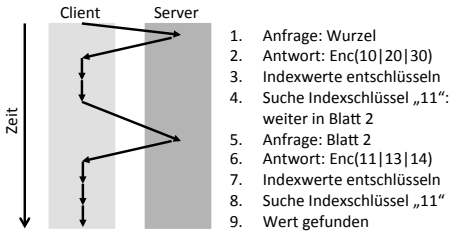

Abb. 2: Indexsuche nach Wert 11 (vgl. [EW16])

5 Tupelweise Auswertung von Nutzer-Präferenzen

ZeroDB bietet keine Möglichkeit zur Verarbeitung von Nutzer-Präferenzen. Bei jedem Datenzugriff soll jedoch ein Zweck zusammen mit der Anfrage angegeben und mit den Präferenzen abgeglichen werden. In [JTD16] wird gezeigt, wie Nutzer-Präferenzen modelliert und gespeichert werden können. Bei Datenbankanfragen steht jedoch die Auswertung von Präferenzen bzgl. der Verwendung einzelner Attribute im Vordergrund. In diesem Abschnitt liegt der Fokus daher auf der Speicherung und Auswertung von Präferenzen für einzelne Nutzerdaten eines Online-Dienstes.

Häufig besitzen Zwecke in Unternehmensabläufen eine hierarchische Abhängigkeit. Damit ein Zweck erreicht werden kann, müssen dafür entweder ein einzelner oder mehrere andere Zwecke erreicht werden. Die Zweckhierarchie eines Online-Dienstes wird durch einen Baum angegeben, wobei jeder Knoten einen Zweck des Online-Dienstes repräsentiert und jede Kante einen übergeordneten Hauptzweck und einen untergeordneten Teilzweck verbindet [BL08]. Abb. 3 zeigt eine beispielhafte Zweck-Hierarchie eines Reiseportals. Darin kann beispielsweise der Zweck *Zusendung von Informationen* per Post oder per E-Mail realisiert werden. Die Überprüfung der Nutzer-Präferenzen für jedes Tupel in einer Antwortmenge einer Datenbankanfrage erhöht den Umfang der Anfragebearbeitung. Der Aufwand zum Zeitpunkt der Anfrage kann reduziert werden, indem die Auswertung vorberechnet wird. In der Tabelle mit den verschlüsselten Werten wird hierfür zu jedem

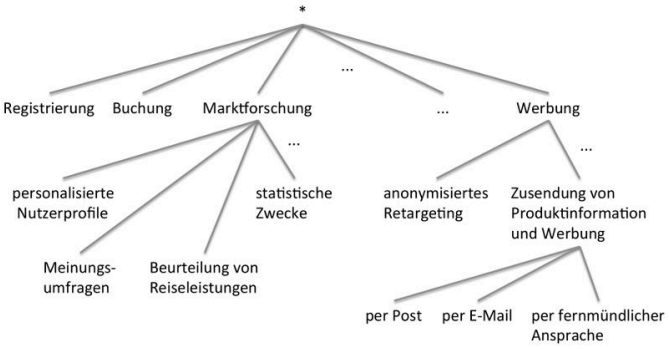

Abb. 3: Zweck-Hierarchien eines Reiseportals

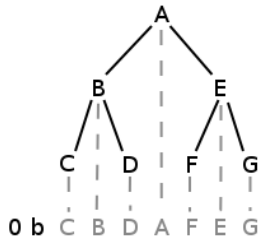

Abb. 4: Transformation einer Zweck-Hierarchien

purpose	code
A	0b0001000
B	0b0100000
C	0b1000000
D	0b0010000
E	0b0000010
F	0b0000100
G	0b0000001

Abb. 5: Zuordnung von Zwecken zu Bitfolgen

Attribut ein weiteres Feld hinzugefügt, welches die von dem jeweiligen Nutzer erlaubten Zugriffszwecke speichert. Zur schnelleren Auswertung dieser Zugriffszwecke, werden sie in Form von Bitfolgen gespeichert.

Die Darstellung der erlaubten Zwecke kann auf eine Bitfolge reduziert werden [BL08], sofern die Menge der Zwecke für eine Tabelle fixiert wurde. In Abb. 4 wird das Konzept dargestellt. Dabei wird jeder Knoten der Hierarchie mit einem Bit assoziiert (Position im String). Es kann für jeden Zweck die Frage beantwortet werden, ob ein Dienst zu genau diesem Zweck auf ein Datum zugreifen darf, daraus ergibt sich für das zugehörige Bit eine 1 (Zugriff gestattet) oder eine 0 (Zugriff verboten). In Abbildung 5 werden Bitfolgen für erlaubte Einzelzwecke dargestellt. Diese lassen sich durch den bitweisen *UND*-Operator leicht kombinierten.

Bei der Anfrage eines Dienstes kann nun für jedes angefragte Attribut eine Bitfolge der gewünschten Zwecke – mithilfe der Zuordnung: Zweck $\rightarrow$ Bit (vgl. Abbildung 4) – ermittelt werden und gegen die gespeicherte Bitfolge verglichen werden. Die Zugriffs-Bitfolge muss dabei komplett in der Attribut-Bitfolge des Wertes „enthalten“ sein. Dazu muss gelten:

$$\textit{Zugriffs-Bitfolge} \;\&\; \textit{Attribut-Bitfolge} = \textit{Zugriffs-Bitfolge}.$$

Hierbei bezeichnet & den bitweisen *UND*-Operator. Diese Eingenschaft lässt sich leicht in der WHERE-Klausel einer SQL-Anfrage formulieren, dadurch kann eine Datenbank effizient die Tupel aus einer Antwort filtern, die mit dem übergebenen Anfragezweck nicht vereinbar sind. Abbildung 6 zeigt ein Beispiel für eine erfolgreiche Anfrage, während die

	Abb. 6: Erfolgsbeispiel	Abb. 7: Fehlerbeispiel 1	Abb. 8: Fehlerbeispiel 2
Zugriffs-Bitfolge	00001010	00001010	00001010
Attribut-Bitfolge	00011110	00010100	00011100
AND	00001010	00000000	00001000
AND=Anfr.	true	false	false

Abbildungen 7 und 8 Beispiele zeigen, bei denen der Zugriff aufgrund des Anfragezwecks fehlschlägt.

Die Bitfolgen zu jedem Attribut eines Tupels können zu einem beliebigen Zeitpunkt vor einer Anfrage berechnet werden – im besten Falle zum Zeitpunkt des Einfügens in die Datenbank – was den Aufwand zum Anfragezeitpunkt stark reduziert. Bei diesem Vorgehen gibt es zwei Nachteile. Um sicherzustellen, dass die Bitfolgen stets aktuell sind, müssen sie bei jeder Änderung von Präferenzen oder der Datenschutzerklärung des Onlinedienstes aktualisiert werden. Dabei ist eine Änderung der Präferenzen von geringem Aufwand begleitet, hier müssen nur die Bitfolgen der Einträge des Individuums geändert werden. Eine Änderung der Datenschutzerklärung hingegen zieht eine Aktualisierung aller Bitfolgen der Tabelle nach sich. Bei sich häufig ändernden Datenschutzerklärungen wäre das Verwenden der Bitfolgen daher kritisch zu betrachten.

Die Anfragebearbeitung mittels Bitfolgen wurde bereits in [BL08] ausführlich evaluiert, daher verweisen wir für eine Übersicht auf die dort gewonnenen Erkenntnisse. Zusammenfassend können wir festhalten, dass sich mit dem gewählten Ansatz Nutzer-Präferenzen effizient auswerten lassen und dass dabei der Zusatzaufwand in einem akzeptablen Rahmen bleibt.

6 Übersicht der Anfragebearbeitung

Dieser Abschnitt beschreibt das Gesamtsystem. Insbesondere wird dabei auf das Zusammenspiel der zuvor vorgeschlagenen Komponenten bei der Beantwortung von Bereichsanfragen auf Attributwerte eingegangen. In Abschnitt 2 wurde bereits festgestellt, dass bei zweckgebundenen Punktanfragen auf einzelne verschlüsselte Tupel mittels Tupel-ID die Nutzerpräferenzen leicht überprüft werden können. Durch die Angabe einer Menge an Tupel-IDs bei einer Datenbankanfrage können daher ebenfalls für die resultierende Antwort tupelweise die Nutzerpräferenzen ausgewertet werden. Wird dabei die im letzten Abschnitt vorgestellte Speicherung von Bitfolgen verwendet, entsteht dadurch nur ein geringer zeitlicher Zusatzaufwand.

Bei einer Bereichsanfrage wird auch eine Menge von Tupeln ausgewählt. Die Herausforderung bei verschlüsselten Daten liegt darin, zu entscheiden welche Tupel in dem gesuchten Bereich liegen. Für diese Aufgabe kann der Index aus der ZeroDB verwendet werden. Wenn ein neues Tupel mit der ID t_{neu} samt Präferenzen für Attributnutzungen in die Nutzerdatenbank geschrieben wird, speichert der verschlüsselte Index gleichzeitig den Wert für das zu indizierende Attribut aus dem Tupel zusammen mit t_{neu} ab. Der so erstellte Index kann nun benutzt werden, um für einen gesuchten Bereich alle Tupel-IDs zu finden. Für diese Art der Datenspeicherung werden zwei Komponenten benötigt.

Abbildung 9 illustriert eine für Bereichsanfragen geeignete Systemarchitektur. Die ZeroDB enthält nur die Indizes für vorab festgelegte Attribute. Die zweite Komponente speichert die Nutzerdaten und -Präferenzen. Sie basiert auf einer angepassten PostgreSQL-Datenbank. Auf der Client-Seite läuft ein spezieller Treiber, der sich um die Anfrageadaption sowie um die benötigten Einfüge-, Aktualisierungs- und Lösch-Operationen in der jeweiligen Daten-

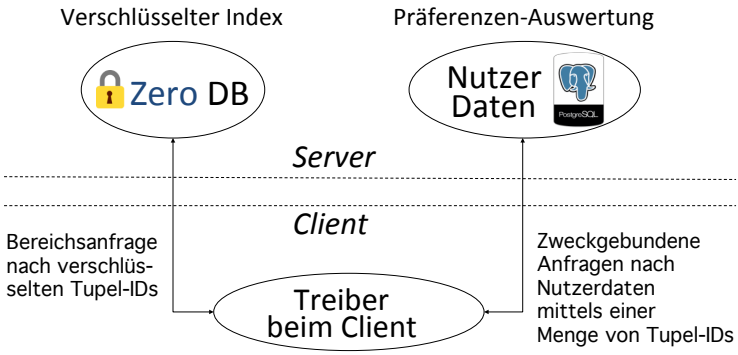

Abb. 9: Systemarchitektur für Anfragen an verschlüsselte Nutzerdaten in der Cloud

bank kümmert. Eine Bereichsanfrage auf ein Attribut wird in zwei Schritten ausgeführt. Zuerst stellt der Treiber eine Anfrage an die ZeroDB, mit der alle Tupel-IDs aus diesem Bereich gefunden werden. Im zweiten Schritt werden aus der PostgreSQL-Datenbank die entsprechenden Tupel angefragt. Die Tupel in der Antwortmenge enthalten jedoch nur Daten, für die ein zweckbezogener Zugriff durch entsprechende Nutzer-Präferenzen autorisiert wurde.

7 Zusammenfassung und Ausblick

Als Ergebnis dieser Arbeit steht die Erkenntnis: Die Aufgabe Bereichsanfragen auf verschlüsselte Daten in unsicheren Cloud-Datenbanken ausführen zu können, lässt sich durch die geschickte Kombination eines verschlüsselten Indexes in Form eines B-Baums in der ZeroDB und eines effizienten Verfahrens zur Auswertung von durch Bitfolgen kodierte Nutzerpräferenzen bewerkstelligen. Dabei kann zu keinem Zeitpunkt der Klartextwert eines Nutzerattributes in der Cloud gelesen werden. Im Gegensatz zu den meisten bisher existierenden Verfahren, die auf schwächere kryptographische Algorithmen zurückgreifen, bleiben die Nutzerdaten als auch der Index mit dem präsentierten Ansatz immer vollständig probabilistisch verschlüsselt.

Wir haben uns in dieser Arbeit darauf beschränkt das System zu motivieren und einzuführen. Daher fehlen Performanzmessungen, die genauer untersuchen, wie gut das System bei einer großen Anzahl von Nutzern funktioniert. Für bis zu 10000 Datensätzen ließ sich in unseren bisherigen Experimenten praktikable Anfragezeiten nachweisen. Eine genaue Performanzanalyse wird in einer zukünftigen Arbeit erfolgen.

Als weitere verbleibende Aufgabe müssten neben den Nutzerdaten auch die gespeicherten Bitfolgen für die Nutzer-Präferenzen verschlüsselt gespeichert und ausgewertet werden. Diese Informationen lassen zwar keine direkten Rückschlüsse auf die verschlüsselten Nutzerdaten zu, allerdings könnten einige Nutzer diese Metadaten dennoch als sensibel

einstufen. Da es für den bitweisen *UND*-Operator jedoch effiziente Algorithmen für homomorphe Verschlüsselung gibt, kann das hier vorgeschlagen Verfahren zukünftig dafür umgestellt werden.

Literatur

[Ag02] Agrawal, Rakesh; Kiernan, Jerry; Srikant, Ramakrishnan; Xu, Yirong: Hippocratic Databases. In: Proceedings of the 28th International Conference on Very Large Data Bases. VLDB '02. VLDB Endowment, S. 143–154, 2002.

[Ar13] Arasu, Arvind; Blanas, Spyros; Eguro, Ken; Joglekar, Manas; Kaushik, Raghav; Kossmann, Donald; Ramamurthy, Ravi; Upadhyaya, Prasang; Venkatesan, Ramarathnam: Secure Database-as-a-service with Cipherbase. In: Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data. SIGMOD '13, ACM, New York, USA, S. 1033–1036, 2013.

[BBL05] Bertino, Elisa; Byun, Ji-Won; Li, Ninghui: Privacy-Preserving Database Systems. In (Aldini, Alessandro; Gorrieri, Roberto; Martinelli, Fabio, Hrsg.): Foundations of Security Analysis and Design III: FOSAD 2004/2005 Tutorial Lectures. Springer Berlin Heidelberg, Berlin, Heidelberg, S. 178–206, 2005.

[BL08] Byun, Ji-Won; Li, Ninghui: Purpose based access control for privacy protection in relational database systems. VLDB J., 17(4):603–619, 2008.

[BS11] Bajaj, Sumeet; Sion, Radu: TrustedDB: A Trusted Hardware Based Database with Privacy and Data Confidentiality. In: Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data. SIGMOD '11, ACM, New York, USA, S. 205–216, 2011.

[EW16] Egorov, Michael; Wilkison, MacLane: ZeroDB white paper. CoRR, 2016.

[Ge09] Gentry, C.: Fully homomorphic encryption using ideal lattices. In: Proceedings of the 41st annual ACM symposium on Theory of Computing. ACM, S. 169–178, 2009.

[Go03] Goh, Eu-Jin: Secure Indexes. IACR Cryptology ePrint Archive, 2003:216, 2003.

[JTD16] Janusz, Daniel; Taeschner, Jochen; Dimitrov, Dimitar: Sichere und datenschutzkonforme Benutzerverwaltung als Online-Service. Datenbank-Spektrum, 16(2):157–166, 2016.

[Ne15] Neuhaus, Christian; Feinbube, Frank; Janusz, Daniel; Polze, Andreas: Secure Keyword Search over Data Archives in the Cloud - Performance and Security Aspects of Searchable Encryption. In: Proceedings of the 5th International Conference on Cloud Computing and Services Science. S. 427–438, 2015.

[PM11] Pearson, S.; Mont, M. C.: Sticky Policies: An Approach for Managing Privacy across Multiple Parties. Computer 09/11, 44:60–68, 2011.

[Po11] Popa, Raluca Ada; Redfield, Catherine M. S.; Zeldovich, Nickolai; Balakrishnan, Hari: CryptDB: Protecting Confidentiality with Encrypted Query Processing. In: Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles. SOSP '11, ACM, New York, NY, USA, S. 85–100, 2011.

[RAD78] Rivest, R. L.; Adleman, L.; Dertouzos, M. L.: On data banks and privacy homomorphisms. Foundations of secure computation, 32(4):169–178, 1978.

[Sh05] Shmueli, Erez; Waisenberg, Ronen; Elovici, Yuval; Gudes, Ehud: Designing Secure Indexes for Encrypted Databases. In: Proceedings of the 19th Annual IFIP WG 11.3 Working Conference on Data and Applications Security. DBSec'05, Springer-Verlag, Berlin, Heidelberg, S. 54–68, 2005.

[SWP00] Song, D. X.; Wagner, D.; Perrig, A.: Practical techniques for searches on encrypted data. In: Proceedings of 2000 IEEE Symposium on Security and Privacy. S &P 2000, S. 44–55, 2000.

[WAE11] Wang, Shiyuan; Agrawal, Divyakant; El Abbadi, Amr: A Comprehensive Framework for Secure Query Processing on Relational Data in the Cloud. In: Secure Data Management - 8th VLDB Workshop, SDM 2011, Seattle, WA, USA, September 2, 2011, Proceedings. S. 52–69, 2011.

[YZW06] Yang, Zhiqiang; Zhong, Sheng; Wright, Rebecca N.: Privacy-Preserving Queries on Encrypted Data. In (Gollmann, Dieter; Meier, Jan; Sabelfeld, Andrei, Hrsg.): Computer Security – ESORICS 2006: 11th European Symposium on Research in Computer Security, Hamburg, Germany, September 18-20, 2006. Proceedings. Springer Berlin Heidelberg, Berlin, Heidelberg, S. 479–495, 2006.