Jörn Müller-Quade, Matthias Huber, Tobias Nilges

Daten verschlüsselt speichern und verarbeiten in der Cloud

Neue Anwendungsformen kryptografischer Verfahren erlauben es, personenbezogene Daten über unterschiedliche Vertrauenszonen hinweg verschlüsselt zu verarbeiten und speichern. Die oft als heiliger Gral der Kryptographie bezeichnete voll-homomorphe Verschlüsselung ist theoretisch eine perfekte Lösung für den Datenschutz im Cloud Computing. Für den Einsatz in der Praxis ist sie jedoch bisher deutlich zu aufwändig. Eine Alternative sind Verfahren mit einer an die konkrete Anwendung angepassten Sicherheit.

Die Kryptographie macht Anwendungen, bei denen potentiell nicht vertrauenswürdige Parteien beteiligt sind, überhaupt erst möglich. Einkaufen im Internet und Online-Banking beispielsweise wären leicht angreifbar, wenn Nachrichten zwischen Anwender und Dienstanbieter abgehört oder manipuliert werden könnten.

Auch für neuartige Cloud Computing-Anwendungen bietet die Kryptographie Lösungen, um die Daten vor Ausspähen oder Veränderung zu schützen. Hierbei muss jedoch zwischen jetzt schon praktisch anwendbaren und prinzipiell möglichen, von der Praxis jedoch noch weit entfernten Lösungen unterschieden werden.

Prof. Dr. Jörn Müller Quade

ist Professor am KIT in Karlsruhe, Direktor am FZI Forschungszentrum Informatik und Sprecher von KASTEL, dem Kompetenzzentrum für angewandte Sicherheitstechnologie.

E-Mail: mueller-quade@kit.edu

Matthias Huber

ist Abteilungsleiter am FZI Forschungszentrum Informatik.

E-Mail: mhuber@fzi.de

Tobias Nilges

ist wissenschaftlicher Mitarbeiter am Institut für Theoretische Informatik (ITI) des KIT.

E-Mail: tobias.nilges@kit.edu

Der heilige Gral der Kryptographie

Im Jahr 1977 veröffentlichten Ronald L. Rivest, Adi Shamir und Leonard Adleman das Verschlüsselungsverfahren RSA [RSA78]. Das nach den Erfindern benannte Verfahren war das erste sogenannte asymmetrische Verschlüsselungsverfahren. Es erlaubte erstmals die Verschlüsselung mit einem öffentlich bekannten Schlüssel. So wurde das Problem des Schlüsselaustauschs gelöst, für das sonst vertrauenswürdige Boten oder persönliche Treffen notwendig waren. Bei dem RSA-Verfahren erzeugt jeder Kommunikationspartner einen privaten und einen öffentlichen Schlüssel. Der öffentliche Schlüssel wird beispielsweise über eine Art Telefonbuch bekannt gemacht. Der private, zur Entschlüsselung notwendige Schlüssel, ist nur dem legitimen Empfänger bekannt und kann auch nicht aus dem öffentlichen Schlüssel abgeleitet werden. RSA wird, in abgewandelter Form, beispielsweise bei sicherer Datenübertragung mit TLS oder PGP im Internet eingesetzt.

Rechnen auf verschlüsselten Daten

Die ursprüngliche Form von RSA, das sogenannte Textbook RSA, hatte eine interessante Eigenschaft: Multiplizierte man zwei (mit dem gleichen Schlüssel) verschlüsselte Nachrichten, so war das Ergebnis ein Chiffrat des Produkts der beiden Nachrichten. Der Grund liegt in den mathematischen Strukturen, auf denen das Verfahren aufgebaut ist. Abbildung 1 zeigt, wie die Multiplikation zweier Chiffrate in Textbook RSA funktioniert. Eine Multiplikation zweier Chiffrate bewirkt eine Multiplikation der darin enthaltenen Klartexte.

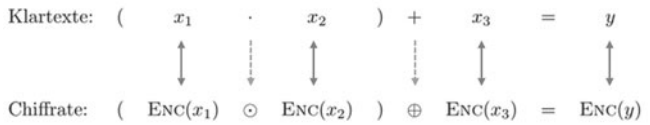
Abbildung 1 | Der Privacy-Homomorphismus in Textbook RSA mit dem öffentlichen Schlüssel e und dem Modulus m: Aufgrund der Eigenschaften der mathematischen Strukturen, in denen gerechnet wird, ergibt das Produkt zweier verschlüsselter Nachrichten eine Verschlüsselung des Produkts beider Nachrichten.

$$\textsc{Enc}(x_1) \cdot \textsc{Enc}(x_1) = x_1^e \cdot x_2^e \mod m = (x_1 \cdot x_2)^e \mod m = \textsc{Enc}(x_1 \cdot x_2)$$

Diese Eigenschaft, auch *Privacy-Homomorphismus* genannt, wurde 1978 entdeckt. Seitdem war es ein Traum vieler Kryptogra-

phen, ein Verschlüsselungsverfahren zu finden, bei dem man Chiffrate sowohl addieren als auch multiplizieren kann und sich dies gleichermaßen auf die wieder entschlüsselten Klartexte auswirkt. Abbildung 2 zeigt eine Beispielrechnung mit Klartexten und Chiffraten für ein fiktives Verfahren mit einem Privacy-Homomorphismus für Addition und Multiplikation, eine sogenannte voll-homomorphe Verschlüsselung.

Abbildung 2 | Ein hypothetisches Verschlüsselungsverfahren mit einem Privacy-Homomophismus, der Addition und Multiplikation erlaubt. Ein solches Verfahren würde über 30 Jahre lang gesucht und von vielen für unmöglich gehalten.



Ein solches Verfahren würde beliebige Berechnungen auf Chiffraten erlauben, weshalb lange danach gesucht wurde. Solch ein Verfahren würde insbesondere das Auslagern von Berechnungen auf geheimen Daten erlauben, wie es für vertrauliches Cloud Computing nötig ist. Es wurde auch als *der heilige Gral der Kryptographie* bezeichnet. Man stellte sich die gesuchte Verschlüsselung wie ein wunderschönes mathematisches Objekt vor. Es wurde aber 30 Jahre lang kein solches Verfahren gefunden, weshalb viele Experten es für unmöglich hielten.

Rechnen mit Schmutz: Die erste voll-homomorphe Verschlüsselung

Craig Gentry veröffentlichte 2009 schließlich die erste voll-homomorphe Verschlüsselung [G09]. Diese war kein wunderschönes mathematisches Objekt, sondern beruhte auf einem schmutzigen Trick.

Lange vor Gentry gab es schon spezielle Verschlüsselungen, bei denen man Chiffrate addieren und multiplizieren konnte. Dies funktionierte allerdings nicht häufig nacheinander. *Schmutzeffekte* führten dazu, dass eine Entschlüsselung nach einer längeren Rechnung nicht mehr möglich war. Die geniale Idee Gentrys war nun, das Rechnen auf Chiffraten so zu verwenden, dass diese Schmutzeffekte korrigiert werden. Eine längere Berechnung auf Chiffraten wird dadurch möglich, dass man eine Rechenoperation auf Chiffraten und anschließend einen Reinigungsschritt durchführt. Mit dem bereinigten, immer noch verschlüsselten Ergebnis kann nun weiter gerechnet werden. Ein Problem dabei ist, dass der Reinigungsschritt selbst wieder eine Berechnung auf Chiffraten ist, und dadurch weiter Schmutz zum Ergebnis hinzufügt. Neben seiner initialen Idee gelang es Gentry auch, den Reinigungsschritt so zu entwerfen, dass er kaum Schmutzeffekte erzeugt und deshalb wirklich reinigend wirkt.

Das Verfahren von Gentry beruht teilweise auf weniger gut untersuchten Sicherheitsannahmen und ist von der Berechnungsgeschwindigkeit nicht für praktische Anwendungen nutzbar. In den letzten Jahren gab es aber gewaltige Fortschritte: Zum einen existieren jetzt Verfahren, die auf gut verstandenen Sicherheitsannahmen basieren, zum anderen konnte die Berechnungsgeschwindigkeit derart verbessert werden, dass sehr spezielle Anwendungen realistisch erscheinen [DM15].

Sicheres Cloud Computing dank voll-homomorpher Verschlüsselung?

Dank voll-homomorpher Verschlüsselungsverfahren lassen sich beliebige Berechnungen auf verschlüsselten Daten durchführen. Dadurch könnten Datensicherheits- und Datenschutzprobleme beim Cloud Computing technisch gelöst werden: Vor dem Auslagern verschlüsselt der Auftraggeber seine Daten mit einem geeigneten Verfahren, der Dienstleister führt die Berechnungen auf den Chiffraten aus und gibt das verschlüsselte Ergebnis, welches nur der Auftraggeber entschlüsseln kann, zurück. Der Dienstleister sieht die Daten nie im Klartext.

Trotz der Fortschritte in den letzten Jahren sind voll-homomorphe Verschlüsselungsverfahren jedoch immer noch weit von einem praktischen Einsatz entfernt. Um ein Programm auf homomorph verschlüsselten Daten auszuführen, muss es, da als Operationen nur Addition und Multiplikation zu Verfügung stehen, als Schaltkreis dargestellt und ausgeführt werden. Da das Ergebnis einer Berechnung auf Chiffraten wieder ein Chiffrat ist, können zur Laufzeit des Programms keine Entscheidungen aufgrund der eigentlichen Daten gefällt werden. Abbruchbedingungen oder bedingte Sprünge aufgrund der verarbeiteten Daten, die bei herkömmlicher Software exzessiv genutzt werden, sind beispielsweise nicht möglich.

Da viel weniger Kontrolloperationen zur Verfügung stehen als bei herkömmlichen Programmen, sind als Schaltkreis implementierte Programme deutlich größer. Dies kann die Ausführung verlangsamen. Außerdem dauert die Auswertung eines einzelnen Schaltkreiselements über tausend mal länger als die entsprechende Auswertung direkt auf den Klartext-Daten. Auch die Datenmengen, die bei voll-homomorpher Verarbeitung entstehen, sind im Verhältnis zu klassischer Verarbeitung gewaltig: mehrere Gigabyte an Daten und Arbeitsspeicher sind selbst bei aktuellen Verfahren nötig. Es ist also für Dienstleister selbst nicht wirtschaftlich, eine solche Technik einzuführen.

Darüber hinaus gibt es für die Datensicherheits- und Datenschutzprobleme beim Cloud Computing eine triviale Lösung: es einfach nicht zu tun und die Berechnungen lokal zu halten. Kryptographische Verfahren für sicheres Cloud Computing werden erst relevant, wenn Cloud Computing mit dem zusätzlichen Aufwand durch Kryptographie immer noch wirtschaftlicher ist, als die Berechnungen mit der eigenen Infrastruktur auszuführen. Bei voll-homomorphen Verschlüsselungsverfahren ist dies derzeit nicht der Fall.

Effizientere Verfahren dank angepasster Sicherheit

In der Kryptographie wird Sicherheit formal in sogenannten *Sicherheitsbegriffen* beschrieben. Diese sind eine exakte mathematische Spezifikation der gewünschten Sicherheitseigenschaft.

Klassische kryptographische Verfahren haben sehr starke Sicherheitsbegriffe. Bei Verschlüsselungen beispielsweise ist die Intuition hinter den dort verwendeten Begriffen, dass Dritte bis auf die Länge des Chiffrats keine weiteren Informationen über den Klartext erlangen dürfen, selbst wenn sie beispielsweise beliebig viele andere Klartext-Chiffrat-Paare kennen. Auch die im vorherigen Abschnitt behandelten voll-homomorphen Verschlüsse-

lungsverfahren erfüllen diese klassischen, sehr starken Sicherheitsbegriffe.

Diese starken Anforderungen haben den Vorteil, dass Verfahren, die diese erfüllen, fast universell eingesetzt werden können. Für bestimmte Anwendungen könnten jedoch schwächere, angepasste Sicherheitsgarantien ausreichen. Auch schließen zu starke Sicherheitsanforderungen effiziente Verfahren unter Umständen aus.

Aus diesem Grund werden in dem Kompetenzzentrum für Angewandte Sicherheitstechnologie (KASTEL) anwendungsspezifische, abgeschwächte Sicherheitsbegriffe untersucht. Diese Anpassung für bestimmte Anwendungen in der Cloud erlaubt einen Kompromiss zwischen Sicherheit und Praktikabilität.

Ein Ergebnis dieser Untersuchungen ist ein Verfahren zum Auslagern von Datenbanken in die Cloud, welches einerseits beweisbare Sicherheit bietet und sich andererseits schon heute praktisch einsetzen lässt. Der Dienstleister lernt dabei zwar bestimmte Informationen über die ausgelagerten Daten. Was genau geheim bleibt und was der Dienstleister erfahren könnte, ist jedoch exakt formal spezifiziert. Ein konkretes Beispiel weiter unten versteckt aus Effizienzgründen nicht die einzelnen Werte in der Datenbank, wohl aber die Relationen zwischen diesen Datenbankeinträgen.

Angepasste Sicherheit für Datenbanken

Das Auslagern von Datenbanken in die Cloud hat vielerlei Vorteile: Pay-per-use-Bezahlmodelle, dynamische Skalierung und eine bessere Ressourcenausnutzung. Darüber hinaus wird die Gefahr eines Datenverlusts dank spezialisierter Anbieter minimiert.

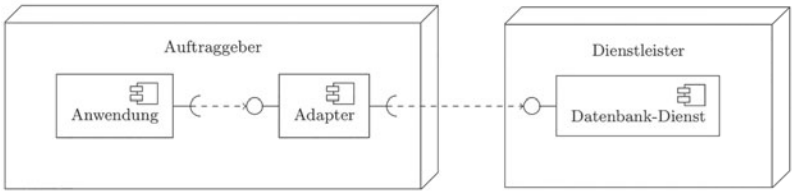
Andererseits gibt es jedoch Herausforderungen für den Datenschutz und die Datensicherheit. Auftraggeber verlieren die Kontrolle über ihre Daten. Beispielsweise dürfen personenbezogene Daten nicht ohne Weiteres einem Dienstleister übergeben werden. Firmengeheimnisse sind zwar nicht zwangsläufig personenbezogen, müssen aber trotzdem vor unberechtigtem Zugriff geschützt werden.

Beim Auslagern von Datenbanken stellt sich insbesondere die Herausforderung, dass die Daten auf Seiten des Dienstleisters effizient durchsuchbar bleiben müssen. Die Datenbank vor dem Auslagern als Ganzes zu verschlüsseln erfüllt zwar die Sicherheitsanforderungen, da der Dienstanbieter die verschlüsselten Daten nicht verwerten kann. Um die Datenbank zu durchsuchen muss sie jedoch komplett heruntergeladen und entschlüsselt werden, was die Idee des Auslagerns ad absurdum führt. Ein Kompromiss zwischen Sicherheit und Effizienz ist notwendig.

Ein Verschlüsselungsadapter für Datenbanken

Ziel ist es also, die Datenbank vor dem Auslagern derart zu verändern und zu verschlüsseln, dass sie auf der Seite des Dienstanbieters zwar effizient durchsucht werden kann, der Dienstanbieter dabei jedoch möglichst wenig über die Daten lernt. Realisiert wird dies mit einem Adapter, der zwischen die Anwendung des Auftraggebers und den Datenbankdienst des Dienstleisters geschaltet wird. Dieser Adapter bereitet die Datenbank zunächst für den Dienstleister auf und verschlüsselt die Inhalte. In der Verwendungsphase übersetzt er die Anfragen an die Datenbank und entschlüsselt die Antworten.

Abbildung 3 | Architektur der Lösung für praktikables Auslagern von Datenbanken. Ein Adapter zwischen Anwendung und Datenbank übernimmt Ver- und Entschlüsselung der Daten, speichert jedoch selbst keine Daten.



Der Adapter selbst speichert hierbei keine Daten und kann beim Auftraggeber selbst oder bei einem vertrauenswürdigen Anbieter laufen.

Wie versteckt man Relationen?

Ziel des Verfahrens ist es nicht, alle Daten vor dem Anbieter zu verstecken. Da auf der Datenbank weiterhin Anfragen effizient ausgeführt werden sollen, benötigt der Dienstleister bestimmte Informationen über die Datenbank. Diese werden ihm in Form von Indextabellen zu Verfügung gestellt. Dadurch lernt der Dienstanbieter zwar, welche Daten in der Datenbank vorhanden sind, nicht aber in welchem Zusammenhang sie zueinander stehen.

Die Tabellen 1 bis 5 zeigen ein Beispiel: In Tabelle 1 sieht man eine Ursprungsdatenbank mit personenbezogenen Informationen. Damit der Dienstanbieter die darin enthaltenen sensitiven Informationen nicht erfährt, darf sie in dieser Form nicht übergeben werden.

Tabelle 1 | Eine Beispieldatenbank mit personenbezogenen Daten

Vorname	Nachname	Krankheit
Peter	Müller	Asthma
Susanne	Müller	Erkältung
Peter	Schmidt	Erkältung

Für jede Spalte in dieser Tabelle werden nun vom Datenbankadapter Indextabellen erstellt. Diese enthalten alle Attributwerte der jeweiligen Spalte sowie eine Liste der Zeilen, in denen der Attributwert in der Originaltabelle auftritt.

Beispielsweise enthält die Indextabelle für das Attribut *Vorname* für den Wert „*Peter*“ die Liste *(1,3)*, da der Vorname Peter in der Originaltabelle in den Zeilen 1 und 3 auftritt.

Tabelle 2 | Indextabelle für das Attribut *Vorname* aus Tabelle 1

values	rows
Peter	*Enc(1,3)*
Susanne	*Enc(2)*

Tabelle 3 | Indextabelle für das Attribut *Nachname* aus Tabelle 1

values	rows
Müller	*Enc(1,2)*
Schmidt	*Enc(3)*

Tabelle 4 | Indextabelle für das Attribut *Krankheit* aus Tabelle 1

values	rows
Asthma	*Enc(1)*
Erkältung	*Enc(1,3)*

Zusätzlich werden diese Listen von Zeilen in den Indextabellen verschlüsselt (dargestellt mit *Enc*). Dadurch sind sie später durch den Dienstanbieter nicht verwertbar. In den Tabellen 2-5 sind verschlüsselte und durch den Dienstanbieter nicht verwertbare Einträge durch kursive Schrift gekennzeichnet.

Tabelle 5 | Die Tabelle 1 nach einer zeilenweisen Verschlüsselung

row	data
1	*Enc(Peter, Müller, Asthma)*
2	*Enc(Susanne, Müller, Erkältung)*
3	*Enc(Peter, Schmidt, Erkältung)*

Neben den Indextabellen wird auch eine sogenannte Datentabelle an den Dienstanbieter übergeben. Diese enthält alle Daten der Ursprungstabelle in verschlüsselter Form. Die Verschlüsselung wird dabei zeilenweise angewandt. Das heißt, jede Tabellenzeile wird separat verschlüsselt (Tabelle 5). Dies erlaubt es, gezielt einzelne Tabellenzeilen abzufragen, wie im nächsten Abschnitt erläutert wird.

Die Tabellen 2 bis 5 werden an den Dienstanbieter übergeben. Dieser kann aufgrund der unverschlüsselten *values*-Spalte in den Indextabellen lernen, welche Werte in der Datenbank auftreten. Durch die verschlüsselten Zeilenlisten und die Datentabelle lernt er jedoch nicht, wie diese Werte zusammenhängen: Die Relationen der Ursprungsdatenbank werden vor dem Dienstleister versteckt. Der Dienstleister lernt zwar beispielsweise, dass jemand mit Vornamen *Peter*, und jemand mit Nachnamen *Schmidt* in der Datenbank vorkommt, er kann jedoch nicht mit Sicherheit sagen, ob *Peter Schmidt* in der Datenbank ist, oder welche Krankheit er hat.

Diese Eigenschaft – Werte bleiben erkennbar, aber Relationen werden geheim gehalten – wurde von uns formal exakt spezifiziert. Damit konnte formal nachgewiesen werden, dass das hier beschriebene Verfahren diese Eigenschaft erfüllt: Die Sicherheit des Verfahrens wurde bewiesen.

Übersetzung von Anfragen

Neben der Ver- und Entschlüsselung der Daten übernimmt der Adapter die Aufgabe der Übersetzung der Datenbankanfragen des Auftraggebers.

Dank des Adapters kann der Auftraggeber auf die beim Dienstleister ausgelagerte Datenbank (Tabellen 2 bis 5) so zugreifen, als würde er auf die Ursprungsdatenbank zugreifen. Der Adapter übersetzt dabei die Anfragen in ein interaktives Protokoll mit der ausgelagerten Datenbank.

Fragt der Auftraggeber beispielsweise nach dem Eintrag von *Peter Müller*, so fragt der Adapter nach dem Eintrag für *Peter* in der Indextabelle für die Vornamen (Tabelle 2) und nach dem Eintrag für *Müller* in der Indextabelle für die Nachnamen (Tabelle 3). Auf diese beiden Anfragen erhält der Adapter die folgenden beiden Antworten:

Peter, *Enc*(1,3)
Müller, *Enc*(1,2)

Der Adapter kann die Listen von Zeilen in diesen beiden Antworten entschlüsseln und berechnet so, dass sich der Eintrag von Peter Müller in Zeile 1 befinden muss. Diese Zeile erhält der Adapter aus der zeilenweise verschlüsselten Datentabelle (Tabelle 5):

1, *Enc*(Peter,Müller,Asthma)

Der Adapter entschlüsselt nun den zweiten Eintrag der zurückgelieferten Zeile und erhält so das Ergebnis der Anfrage des Auftraggebers.

Der Preis der Sicherheit

Das obige Beispiel zeigt, dass eine einzelne Anfrage durch mehrere Anfragen ersetzt werden muss. Dadurch entsteht, neben der Entschlüsselung, ein zusätzlicher Aufwand, mit dem man sich die Datensicherheit erkauft.

Für den Betrieb von Datenbanksystemen ist es wichtig, wie sie sich bei großen Datenbeständen verhalten. Die Geschwindigkeit, mit der Anfragen beantwortet werden können, wird im sogenannten *O-Kalkül* zum Verhältnis der Datenbankgröße gesetzt. Man will wissen, wie gut das System bei größer werdenden Datenmengen skaliert. Bei herkömmlichen Datenbanksystemen hängt die Zeit, die notwendig ist, um eine Anfrage zu beantworten, vom Logarithmus der Größe der Datenbank ab.

Das hier beschriebene Verfahren ist kein neues Datenbanksystem, sondern eine Methode, die bestehende Datenbanken geschickt nutzt. Die zur Beantwortung einer Anfrage auf eine einzelne Tabelle (Indextabelle oder Datentabelle) notwendige Zeit hängt dadurch ebenfalls von der Größe der Ursprungstabelle ab. Das Verhalten ist also bei wachsenden Datenmengen wie bei herkömmlichen Datenbanksystemen: Die Beantwortung von Anfragen ist zwar aufwändiger, der Aufwand steigt aber bei großen Datenmengen prinzipiell ähnlich zu dem herkömmlicher Datenbanksysteme.

Anwendung in der Praxis

In den Projekten MimoSecco[1] und Cumuls4j[2] wurde das hier beschriebene Verfahren angewandt [AGH11, HGSB13]. Praxistests haben gezeigt, dass das Verfahren bereits heute praktisch eingesetzt werden kann.

Die in Cumulus4j entstandene Implementierung ist frei verfügbar.[3] Cumulus4j ist eine Erweiterung für die in Java geschriebene Datenbankabstraktionsschicht DataNucleus[4].

Im letzten Abschnitt wurde gezeigt, dass das Verfahren prinzipiell ähnlich skaliert wie herkömmliche Datenbanksysteme. Bei konkreten Anwendungen ist für die Wirtschaftlichkeit einer Verschlüsselungslösung der absolute Zusatzaufwand jedoch ausschlaggebend. Über diesen lassen sich nur schwer pauschale Aus-

1 https://sites.google.com/site/mimoseccoproject/
2 http://cumulus4j.fzi.de/
3 http://www.datanucleus.org/
4 http://www.cumulus4j.org/latest-dev/de/

sagen treffen, da er von in der Datenbank gespeicherten Daten sowie von den an die Datenbank gestellten Anfragen abhängt. Durch daten- und anfragespezifische Optimierungen ist es jedoch möglich, diesen Aufwand zu minimieren.

Fazit

Kryptographische Berechnungen bieten hohe Vertraulichkeitszusagen. Im Fall der voll-homomorphen Verschlüsselung ist der Aufwand allerdings so hoch, dass ein praktischer Einsatz bisher nicht sinnvoll ist. Bereits heute können Daten vertraulich in der Cloud gespeichert werden. Der in diesem Beitrag vorgestellt Vorschlag versteckt nicht die komplette Datenbank vor dem Dienstleister, sondern nur die in der Datenbank enthaltenen Relationen. Er kann daher einzelne Werte aus der Datenbank erfahren.

Der Aufwand könnte für den Auftraggeber weiter verringert werden, wenn dieser den Adapter von einem Dienstleister betreiben ließe. Dies wäre dann beispielsweise als Auftragsdatenverarbeitung zu gestalten. Es bleibt aber zu klären, ob die verschlüsselte Verarbeitung zu datenschutzrechtlichen Erleichterungen für den Auftraggeber führt. Dürften die Daten mit geringeren Auflagen bei Dienstleistern verarbeitet werden, als dies für konventionelle Verarbeitung nach § 11 BDSG in Auftragsdatenverarbeitung oder nach den weitreichenderen Regelungen in Drittländern möglich ist?[5] Dann könnte die verschlüsselte Verarbeitung die Rahmenbedingungen für Cloud-Anwendungen mit personenbezogenen Daten vereinfachen.

Literatur

[AGH11] Achenbach, Dirk; Gabel, Matthias; Huber, Matthias: *MimoSecco: A Middleware for Secure Cloud Storage Improving Complex Systems Today*, Advanced Concurrent Engineering, Springer London 2011

[DM15] Ducas, Léo; Micciancio, Daniele: FHEW: *Bootstrapping Homomorphic Encryption in Less Than a Second*, Advances in Cryptology - EUROCRYPT 2015, Springer Berlin Heidelberg 2015

[G09] Gentry, Craig: *A Fully Homomorphic Encryption Scheme*, Stanford University, Stanford, CA, USA 2009

[HGSB13] Huber, Matthias; Gabel, Matthias; Schulze, Marco; Bieber, Alexander: *Cumulus4j: A Provably Secure Database Abstraction Layer*, Security Engineering and Intelligence Informatics - CD-ARES 2013, Workshops: MoCySEn and SeCIHD, Lecture Notes in Computer Science, Springer, Germany 2013

[RSA78] Rivest, Ron; Shamir, Adi; Adleman, Leonard: *A Method for Obtaining Digital Signatures and Public-key Cryptosystems*, Commun. ACM, ACM, New York, NY, USA, 1978

5 Zu Aspekten dieser Frage nehmen in diesem Heft die Aufsätze von R. Steidle/U.Pordesch und M. Knopp Stellung.

Springer Gabler

springer-gabler.de

Big Data: Chancen und Herausforderungen für Unternehmen

Joachim Dorschel (Hrsg.)
Praxishandbuch Big Data
2015. 364 S. 40 Abb. Geb.
€ (D) 59,99 | € (A) 61,67 | * sFr 75,00
ISBN 978-3-658-07288-9 (Print)
€ 46,99 | * sFr 60,00
ISBN 978-3-658-07289-6 (eBook)

- Alles, was Unternehmer über Big Data wissen müssen
- Best Practices geben Einblick in die Anwendung von Big Data in verschiedenen Branchen wie Banken, Telekommunikation, Industrie
- Schwerpunkt Datenschutz und Datensicherheit im Unternehmen

€ (D) sind gebundene Ladenpreise in Deutschland und enthalten 7% MwSt. € (A) sind gebundene Ladenpreise in Österreich und enthalten 10% MwSt. Die mit * gekennzeichneten Preise sind unverbindliche Preisempfehlungen und enthalten die landesübliche MwSt. Preisänderungen und Irrtümer vorbehalten.

Jetzt bestellen: springer-gabler.de

A13818