H. Langweg, M. Meier, B.C. Witt, D. Reinhardt (Hrsg.): Sicherheit 2018,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2018 221

Homomorphe Verschlüsselung für Cloud-Datenbanken: Übersicht und Anforderungsanalyse

Lena Wiese[1], Daniel Homann[1], Tim Waage[1], Michael Brenner[2]

Abstract:

Die Auslagerung von Daten in Cloud-Datenbanken verspricht eine Reihe von Vorteilen wie reduzierte Wartungskosten, Flexibilität der Ressourcenverteilung und einfachen, ortsunabhängigen Zugriff. Diese Datenbanken bieten dabei eine Vielzahl von Funktionalitäten, um Berechnungen auf Daten auszuführen. Datensicherheit (einschließlich dem Schutz persönlicher Daten) ist in Cloud-Datenbanken jedoch noch nicht angemessen umgesetzt worden. Konventionelle Verschlüsselungsverfahren garantieren zwar hohe Sicherheit während Transport und Speicherung, verhindern aber auch weitere Berechnungen auf den Daten. Modernere homomorphe Verschlüsselungsverfahren versprechen dagegen sowohl Datensicherheit als auch die Möglichkeit, auf verschlüsselten Daten zu rechnen. Das bestehende System *FamilyGuard* kombiniert bisher eigenschaftsbewahrende Verschlüsselungsverfahren. Um die Funktionalität auf Aggregationsfunktionen zu erweitern, soll in Zukunft auch homomorphe Verschlüsselung eingesetzt werden. In diesem Artikel geben wir eine Übersicht über die verschiedenen Arten homomorpher Verschlüsselungsverfahren und ihre Sicherheitsgrundlagen. Im Anschluss stellen wir Anforderungen für den Einsatz homomorpher Verfahren in Cloud-Datenbanken auf.

Keywords: Homomorphe Verschlüsselung; Datenbanken-Sicherheit

1 Einleitung

Moderne Clouddienste ermöglichen das effiziente Verwalten großer Datensätze. Mit solchen Diensten kann zudem zur Datenanalyse moderne Hochperformanz-Hardware als Platform- oder Software-as-a-Service genutzt werden. Datenbanken können zum Einen Daten zuverlässig speichern und zum Anderen Berechnungen auf den gespeicherten Daten ausführen. Auch Databases-as-a-Service werden von Dienstanbietern in der Cloud angeboten. Jedoch ist über die Vertrauenswürdigkeit der Dienstanbieter oft kein abschließendes Urteil möglich. So ist es teilweise unklar, in welchem Land die Daten gespeichert werden. Der Einsatz von Datenverschlüsselung wird daher aus Datenschutzgründen notwendig.
Es existieren starke kryptographische Verfahren, um den Schutz vertraulicher Daten beweisbar sicherzustellen. Eine naive Lösung des Datenschutzproblems wäre es also,

[1] Universität Göttingen, Institut für Informatik, Goldschmidtstraße 7, 37077 Göttingen, Germany, E-Mail: {wiese, homann, waage}@cs.uni-goettingen.de
[2] Universität Hannover, Fachgebiet Computational Health Informatics, 30159 Hannover, Germany, E-Mail: brenner@luis.uni-hannover.de

doi:10.18420/sicherheit2018_17

die Daten mit einem konventionellen symmetrischen Verschlüsselungsverfahren (wie AES) zu verschlüsseln. Ein erheblicher Nachteil ist jedoch, dass bei einem Lesezugriff zwecks Entschlüsselung alle Daten wieder zum Datenbesitzer transferiert werden müssen. Schreibzugriffe führen darüber hinaus zu weiteren Verschlüsselungsschritten und Zugriffe durch mehrere Besitzer erfordern zusätzlichen Aufwand für eine Schlüsselverwaltung. Neben dem Vertraulichkeitsschutz gilt es daher auch die Daten effizient verarbeiten zu können. Eine Option ist die Entwicklung spezieller kryptographischer Verfahren, mit denen verschlüsselte Daten ausgewertet werden können.
Ein konkreter Anwendungsfall ist die Analyse medizinischer Datensätze in der Cloud (zum Beispiel in der personalisierten Medizin). Solche Datensätze und Clouddienste existieren bereits (wie zum Beispiel unter `http://www.biobanken.de`) und ihre Zahl wächst stetig – besonders für personalisierte Diagnosen oder Behandlungen [Sh17]. Die Vertraulichkeit personenbezogener Daten ist in medizinischen Anwendungen extrem wichtig – etwa bei der Ausführung medizinischer Analysen (zum Beispiel die Analyse von Kohorten ähnlicher Patienten). Die Auswertung von medizinischen Daten in Clouddiensten erfordert jedoch neue Sicherheitsverfahren, um den Schutz von Patientendaten zu gewährleisten.

2 Hintergrund

2.1 Verschlüsselungsverfahren

Der Schutz der Daten und ihre Nutzbarkeit sind in der Regel zwei gegensätzliche Anforderungen. Kommt es zur Verarbeitung vertraulicher Daten, sollten eine vollständige Entschlüsselung, Verarbeitung und anschließende Neuverschlüsselung vermieden werden. Stattdessen sollte eine (Vor-)Verarbeitung der verschlüsselten Daten möglich sein, um die gewünschten Analyseergebnisse unter Aufrechterhaltung der Vertraulichkeit zu erhalten. Dies wird durch spezielle Verschlüsselungsverfahren erreicht, mit denen zumindest Teile der Daten verschlüsselt werden, bevor sie in einem Clouddienst gespeichert werden:

- **eigenschaftsbewahrende** Verschlüsselungsverfahren (engl.: property-preserving encryption; PPE), bei denen sich Eigenschaften des Klartextes auf den verschlüsselten Text übertragen. Hier existieren Verfahren zur durchsuchbaren und ordnungsbewahrenden Verschlüsselung, so dass eine Suche nach verschlüsselten Suchworten und eine Sortierung verschlüsselter Daten möglich ist.
- **homomorphe** Verschlüsselungsverfahren (engl.: homomorphic encryption; HE), mit denen eine Auswertung auf verschlüsselten Daten zu einem verschlüsselten Ergebnis führt, das der anfragende Benutzer dann erhält und entschlüsselt. Der Clouddienst kann also weder die gespeicherten Daten noch das Ergebnis selbst entschlüsseln.

2.2 Einsatz von Verschlüsselung in Datenbanken

Im Projekt FamilyGuard wurden erstmalig mehrere durchsuchbare und ordnungsbewahrende Verschlüsselungsverfahren in einem einheitlichen Rahmen implementiert, vergleichend mit den Spaltenfamilien-Datenbanken HBase und Cassandra getestet und ihre Sicherheitseigenschaften analysiert [WW17]. Spaltenfamilien-Datenbanken gehören zur Gruppe der nichtrelationalen („NoSQL“) Datenbanken [Wi15]. Sie werden wegen ihres flexiblen und effizienten Verhaltens gerne von Cloudspeicher-Anbietern verwendet, bei denen verschiedene Kunden Speicherplatz mieten können. Dies führt zu Bedenken, wie vertrauliche Daten vor einem neugierigen Cloudspeicher-Anbieter oder anderen Angriffen von unbekannten Dritten geschützt werden können. Bisher bieten Spaltenfamilien-Datenbanken keine Methoden an, mit denen der Schutz vertraulicher Daten vor unbefugtem Zugriff sichergestellt werden kann. Das Hauptziel des FamilyGuard-Projektes war es daher, eine sichere und durch den Benutzer anpassbare verschlüsselte Speicherung von Daten in Spaltenfamilien-Datenbanken zu ermöglichen. Dieses Ziel wurde im Projekt FamilyGuard erreicht, indem Verfahren der durchsuchbaren und ordnungsbewahrenden Verschlüsselung für Spaltenfamilien-Datenbanken nutzbar gemacht wurden. Mit diesen (nun erstmals in Java quelloffen verfügbaren) Verfahren können die Datenbanken die folgenden zwei Funktionalitäten anbieten:

1. verschlüsselte Daten können nach verschlüsselten Suchworten durchsucht werden;
2. verschlüsselte numerische Daten können sortiert werden und damit auch Bereichsabfragen (nach Intervallen von Werten) beantwortet werden.

Aus den existierenden Verfahren zur durchsuchbaren und ordnungsbewahrenden Verschlüsselung konnten solche Schemata identifiziert und implementiert werden, die den Anforderungen an den praktischen Einsatz in den unmodifizierten Datenbanksystemen entsprechen und die jeweils bestmöglichen Sicherheitseigenschaften in ihrer Kategorie aufweisen. Eine umfassende Implementierung verschiedener geeigneter Verfahren steht bereits als quelloffenes Framework unter `https://github.com/dbsec/FamilyGuard/` zur Verfügung. Die Performanz konnte sowohl mit synthetischen Benchmarks als auch mit praxisorientierten Messungen quantifiziert werden. In zukünftigen Arbeiten sollen homomorphe Verschlüsselungsverfahren in das bestehende System eingebunden und mit den bereits vorhandenen Verfahren (also durchsuchbaren und ordnungsbewahrenden Verschlüsselungsverfahren) kombiniert werden. Unsere Vision ist eine „verschlüsselte Datenspeicherung als Dienst“ für die Spaltenfamilien-Datenbanken. Auf diese Weise kann der Kunde dann mit der Cloud-Datenbank vielseitig interagieren:

- zum Einen kann er Berechnungen auf den gespeicherten Daten ausführen
- zum Anderen lassen sich diese Berechnungen (auf Grundlage der homomorphen Verschlüsselung) kombinieren mit Selektionsanfragen (auf Grundlage der bereits vorhandenen durchsuchbaren und ordnungsbewahrenden Verschlüsselung)

Andere Systeme bieten diese Funktionalität nur eingeschränkt an. Als eine alternative Implementierung, die sowohl homomorphe als auch eigenschaftsbewahrende Verfahren in Cloud-Datenbanken einsetzt, bietet das Projekt CryptDB [Po12; PZB15] eine sogenannte SQL-bewusste Verschlüsselung. An homomorphen Verfahren werden aber nur die beiden einfachsten Verfahren zu rein additiv-homomorpher Verschlüsselung (Paillier [Pa99]) und zu rein multiplikativ-homomorpher Verschlüsselung (ElGamal [Ga84]) implementiert. Das bedeutet, dass nur eingeschränkte Funktionen (nur Addition oder nur Multiplikation) auf den verschlüsselten Daten berechnet werden können; es können so keine generellen Funktionen berechnet werden, die eine Kombination von Addition und Multiplikation benötigen. Außerdem kann kein symmetrisches Entschlüsselungsverfahren berechnet werden, wie es für die delegierte Verschlüsselung (siehe Abschnitt 3.4) nötig ist. Monomi [Tu13] erweitert CryptDB um mehr SQL-Anfragen, die nun ausgeführt werden können – jedoch mit höherem Rechenaufwand und Speicherplatzbedarf auf Kundenseite (statt auf Cloudseite).
Weitere Ansätze verlangen teure kryptographische Hardware auf Seiten der Cloud-Datenbank (wie etwa TrustedDB [BS14] oder Cipherbase [Ar13]). Es findet dann eine Entschlüsselung der Daten auf Seiten der Cloud-Datenbank innerhalb der kryptographischen Hardware statt und der Cloudanbieter benötigt dazu immer den geheimen Schlüssel des Benutzers; daher sind diese Hardware-basierten Ansätze für eine datenschutzkonforme Analyse ungeeignet.

2.3 Einsatz von Verschlüsselung zur Programmausführung

Neben der etablierten Transportverschlüsselung sollen durch den Einsatz von Verschlüsselung während der Programmausführung auch Sicherheitsmechanismen während der aktiven Datenverarbeitung auf nichtvertrauenswürdigen Ressourcen ermöglicht werden. Dazu werden zur Zeit sowohl einzelne Techniken des verschlüsselten Rechnens, wie homomorphe Kryptographie, Secure Multiparty Computation oder Secure Function Evaluation.
Im Projekt *hcrypt* (`http://hcrypt.com`) ist ein verschlüsseltes Prozessormodell entstanden, mit dem beliebige, verschlüsselte, nicht-sequenzielle Programme mit freiem Lese- und Schreibzugriff auf verschlüsselten Speicher ausgeführt werden können [PBS11]. Diese generische Rechenmaschine kann zwar alle möglichen Programme ausführen, sie unterliegt aufgrund der notwendigen sequentiellen Abarbeitung der rein homomorph verschlüsselten Schaltkreisgatter erheblichen Laufzeitbeschränkungen.
Während dieses Modell die Wirksamkeit der Verschlüsselung zur Laufzeit belegt, offenbart es zugleich die Notwendigkeit der Effizienzsteigerung, um praktische Anwendbarkeit zu erlangen. Dies kann grundsätzlich auf zwei verschiedene Arten erfolgen: Zum Einen basiert die Performanz des Modells auf der Leistungsfähigkeit der Implementierung der verwendeten homomorphen Verschlüsselung. Das heißt, dass sich Fortschritte in diesem Bereich automatisch auf die Effizienz der Anwendungen auswirken. Zum Anderen kann die Effizienz durch den Einsatz geschickt arrangierter Protokolle wesentlich verbessert werden, bei denen konstruktive Maßnahmen ergriffen werden [BS13] oder bei denen nicht die gesamte Verarbeitung auf homomorpher Kryptografie beruht und hybrid mit anderen Technologien verbunden wird.

3 Homomorphe Verfahren

Wir geben nun eine Übersicht über Varianten der homomorphen Verschlüsselung. Tabelle 1 listet die der Sicherheit der Verfahren zugrundeliegenden mathematischen Probleme auf.

3.1 Additiv-homomorphe Verschlüsselung

Eine additiv-homomorphe Verschlüsselungsfunktion *Enc* hat die Eigenschaft, dass sich zu zwei Schlüsseltexten *Enc(a)* und *Enc(b)* effizient die verschlüsselte Summe *Enc(a + b)* berechnen lässt. Das grundlegende Verfahren zur additiv-homomorphen Verschlüsselung ist das Paillier-Verfahren [Pa99], das auf der Zusammengesetze-Reste-Annahme basiert. Es gibt zudem folgende neuere additiv-homomorphe Verfahren:

- Das Joye-Libert-Verfahren [JL13] basiert auf dem traditionellen Goldwasser-Micali-Verfahren [GM84], hat aber eine höhere Effizienz bei der Entschlüsselung. Das Verfahren ist sicher unter der Quadratische-Reste-Annahme.
- Das Castagnos-Laguillaumie-Verfahren [CL15] ist sicher unter der Diffie-Hellman-Annahme.
- Das Fousse-Lafourcade-Alnumaimi-Verfahren [FLA11] ist sicher unter der Höhere-Reste-Annahme.

3.2 Multiplikativ-homomorphe Verschlüsselung

Eine multiplikativ-homomorphe Verschlüsselungsfunktion hat die Eigenschaft, dass sich zu zwei Schlüsseltexten *Enc(a)* und *Enc(b)* effizient das verschlüsselte Produkt *Enc(a × b)* berechnen lässt. Das grundlegende Verfahren zur multiplikativ-homomorphen Verschlüsselung ist das ElGamal-Verfahren [Ga84] und seine auf elliptischen Kurven basierende Variante [Ko87]. Zudem gibt es als neueres rein multiplikativ-homomorphes Verfahren das Desmedt-Gennaro-Kurosawa-Shoup-Verfahren [De10]: es ist sicher unter der Diffie-Hellman-Annahme und baut auf dem Cramer-Shoup-Verfahren [CS98] auf.

3.3 Vollhomomorphe Verschlüsselung

Die Idee beliebiger Kombinationen von Additionen und Multiplikationen auf verschlüsselten Daten wurde bereits 1978 von Rivest, Adleman und Dertouzos beschrieben [RAD78]. Es gab jedoch lange kein Verfahren (auch keine theoretische Konstruktion) eines solchen vollhomomorphen Verfahrens. Aus diesem Grund und aus Effizienzgründen wurden begrenzt („somewhat“) homomorphe Verfahren entwickelt. In diesen Verfahren ist die Anzahl der

Additionen in der Regel unbeschränkt jedoch die Anzahl der Multiplikationen begrenzt. Erst im Jahre 2009 wurde von Craig Gentry ein theoretisches vollhomorphes Verschlüsselungsverfahren vorgeschlagen. Weil nun die logischen Operatoren XOR und AND auf verschlüsselten Daten ausgeführt werden können, ist es möglich beliebige Operationen beliebig oft auf den verschlüsselten Daten zu berechnen. Die Sicherheit von Gentry's erster Konstruktion [Ge09; Ge10] eines vollhomomorphen Verfahrens basiert auf der Komplexität zweier verschiedener Probleme. Zum einen basiert die Sicherheit der von ihm vorgeschlagenen homomorphen Verschlüsselung auf der Komplexität des Shortest Independent Vector Problems (SIVP) in von Idealen erzeugten Gittern. Zum anderen basiert die Sicherheit des von Gentry entworfenen Bootstrapping-Algorithmus auf der Komplexität des Sparse Subset Sum Problems. Grundsätzlich liegt ein wesentliches Problem der homomorphen Multiplikation darin, dass jeder Multiplikationschritt das Ergebnis verfälscht: es treten Ungenauigkeiten durch Rauschen (engl.: noise) im Berechnungsergebnis auf, die sich mit jeder Multiplikation verstärken. Dadurch ist die Anzahl der ausführbaren Multiplikationen begrenzt, wenn man ein korrektes Ergebnis erhalten will. Diese Ungenauigkeit des Ergebnisses wird durch das Gentry-Verfahren vermieden, indem ein Auffrischen der verschlüsselten Daten nach einer gewissen Anzahl von Multiplikationen durchgeführt wird; dieses Auffrischen nennt Gentry „bootstrapping". Da dieses Auffrischen sehr zeitaufwändig ist, wurden zahlreiche Optimierungen [Di10] vorgeschlagen.
Aktuell können mit der sogenannten nivellierten homomorphen Verschlüsselung (engl.: leveled homomorphic encryption) die Parameter so optimiert werden, dass eine gewählte Anzahl von Multiplikationen ausgeführt werden kann, ohne dass die Ungenauigkeiten zu groß werden [BGV14]. Derzeit gilt das Fan-Vercauteren-Verfahren [FV12] als das effizienteste bei den höchsten Sicherheitseigenschaften. Die Sicherheit dieses Algorithmus basiert auf dem Ring Learning With Errors (RLWE) Problem; es optimiert das Verfahren von Brakerski [Br12], das auf dem Learning with Errors (LWE) Problem basiert.
Ein weiteres nivelliertes Verfahren ist das YASHE-Schema [Bo13]. Sein Sicherheitsbeweis basiert auf der NTRU-Annahme [HPS98], die besagt, dass es schwer ist in einem speziellen Verband einen kürzesten Vektor zu finden, beziehungsweise zu einem nicht am Verband beteiligten Vektor den nächstliegenden Vektor im Verband zu finden. Jedoch basiert das Schema auf einer überdehnten (engl.: „overstretched") Variante der NTRU-Annahme, für die bereits verschiedene Angriffe entdeckt wurden [ABD16; CJL16; KF16].
Zahlreiche Verfahren bauen auf diesen grundlegenden Schemata auf, um weitergehende Funktionalitäten zu ermöglichen – so zum Beispiel auch Mehrbenutzerverfahren, die verschiedene Benutzer-Schlüssel unterstützen [LTV17].

3.4 Delegierte Verschlüsselung

Um den Nutzer von der langwierigen vollhomomorphen Verschlüsselung zu entlasten, wurde ein Verfahren vorgeschlagen, um die homomorphe Verschlüsselung an einen Cloudanbieter zu delegieren ohne Klartextdaten preiszugeben. Der Kunde muss nur eine schnelle symmetrische Verschlüsselung (z.B. AES) auf seinen Daten ausführen und die so verschlüsselten

Name (Englisch)	Beschreibung
Shortest Independent Vector	Gegeben ein n-dimensionales Gitter L und eine Konstante $\gamma > 1$. Bestimme linear unabhängige Vektoren $v_1, \ldots v_n \in L$, sodass $\max \|v_i\| < \gamma\lambda(n)$. Dabei ist $\lambda(n)$ die minimale Norm von n unabhängigen Vektoren aus L.
Sparse Subset Sum	Bestimme $A' \subset A$, sodass $\sum_{a \in A'} a \equiv t \mod M$ für $A \subset \mathbb{N}$ und $t, M \in \mathbb{N}$.
Learning With Errors	Bestimme eine lineare Funktion $f : \mathbb{Z}_q^n \to \mathbb{Z}_q$, sodass $y = f(x)$ mit hoher Wahrscheinlichkeit für Stichproben (x, y) mit $x \in \mathbb{Z}_q^n$ und $y \in \mathbb{Z}_q$ gilt.

Tab. 1: Den homomorphen Verschlüsselungs-Verfahren zugrundeliegende mathematische Probleme

Daten dann in der Cloud-Datenbank speichern. Nur der kurze symmetrische Schlüssel muss auf Kundenseite dann noch homomorph (mit dem öffentlichen Schlüssel des Kunden) verschlüsselt und dem Clouddienst zur Verfügung gestellt werden. Der Clouddienst verschlüsselt dann die Daten ein zweites Mal homomorph (ebenfalls mit dem öffentlichen Schlüssel des Kunden) und führt danach mittels des homomorph verschlüsselten symmetrischen Schlüssels eine Entschlüsselung des symmetrischen Verfahrens durch. Danach liegen die Daten dann allein homomorph verschlüsselt vor, womit die zeitaufwändige homomorphe Verschlüsselung an den Clouddienst delegiert wurde. Bisher entwickelte homomorphe Verfahren arbeiten in Kombination mit AES oder dem leichtgewichtigen SIMON [Ca15; CLT14; GHS12; LN14; NLV11].

Diese Delegation der homomorphen Verschlüsselung ist wichtig für ressourcenarme Endgeräte wie sie zum Beispiel zunehmend für mobile Gesundheitsanwendungen eingesetzt werden. Eine Schnittstelle für AES-verschlüsselte Daten bietet auch generell den Vorteil, das auf Benutzerseite nur konventionelle AES-Verschlüsselung benötigt wird; es müssen dort also keine speziellen Verschlüsselungsmodule installiert werden.

Ein Anwendungsfall für die delegierte Verschlüsselung sind Fitnesstracker: diese Fitnesstracker zeichnen zumeist als Armband kontinuierlich die sportliche Betätigung des Nutzers auf und übermitteln sie an eine Smartphone-App; Smartphones selbst sind nicht performant genug, um eine volle homomorphe Verschlüsselung auf großen Mengen von Messdaten auszuführen. Fitness-Daten können aber mit einem Armband generiert werden und zunächst mittels einer Smartpone-App schneller symmetrisch (etwa mit AES) verschlüsselt werden; anschließend können sie dann auf Seiten der Cloud-Datenbank homomorph verschlüsselt werden (durch delegierte Verschlüsselung) und danach Statistiken (zum Beispiel Wochen- oder Tagesdurchschnitte) in der Datenbank berechnet werden.

3.5 Implementierungen homomorpher Kryptosysteme

Seit der Entdeckung des Verfahrens zur Konstruktion unbegrenzt homomorpher ("fully homomorphic") Kryptosysteme aus der Kombination begrenzter homomorpher ("somew-

hat homomorphic") Kryptosysteme und einem sogenannten Bootstrapping-Mechanismus wurden zunächst Gitter- und Ganzzahl-basierten Verfahren implementiert. Beispiele sind

- M. Brenner, H. Perl, M. Smith (2011): Libscarab; hcrypt-Project `http://hcrypt.com`
- C. Gentry, S. Halevi (2011): Implementing Gentry's Fully Homomorphic Encryption Scheme, Springer LNCS Vol. 6632

Nach einer mehrjährigen Konsolidierungsphase haben sich die RLWE-basierten Systeme durchgesetzt. Bisher wurden Implementierungen unterschiedlichen Umfangs vorgelegt:

- SEAL `http://sealcrypto.org` (Microsoft Research)
- HElib `https://github.com/shaih/HElib` (Shai Halevi / IBM, Victor Shoup)
- NFLlib `https://github.com/CryptoExperts/FV-NFLlib` (Paillier / CryptoExperts)
- Palisade `https://git.njit.edu/groups/palisade` (NJIT)
- cuHE `https://github.com/vernamlab/cuHE` (Worcester Polytechnic Institute)
- HEAAN `https://github.com/kimandrik/HEAAN` (Seoul National University, UCSD)
- TFHE `https://github.com/tfhe/tfhe` (Ilaria Chillotti, Nicolas Gama et al.)

Die tatsächliche Anwendung der genannten Implementierungen unterliegt zur Zeit aufgrund des experimentellen Charakters wesentlichen praktischen Beschränkungen. Insbesondere ist die Interoperabilität derzeit nicht gegeben. Aktuell arbeiten die Gruppen SEAL, HElib, NFLlib, Palisade cuHE, HEAAN und Libscarab zusammen mit Industriepartnern, Universitäten und den US-amerikanischen Behörden NIST und NIH an der Standardisierung homomorpher Verschlüsselungsverfahren (`http://HomomorphicEncryption.org`). Dies umfasst auch eine einheitliche Programmierschnittstelle (API) [Br], um eine Modularität der Programmentwicklung und Austauschbarkeit der Algorithmen zu ermöglichen.

4 Funktionale Anforderungen für den Einsatz homomorpher Verfahren in Cloud-Datenbanken

Obwohl die vollhomomorphen Verschlüsselungsverfahren in der Theorie immer weiter verbessert werden, sind sie in der Praxis immer noch schwer anzuwenden aufgrund der exponentiellen Vergrößerung des Schlüsseltextes gegenüber dem Klartext und der langen Berechnungsdauer. Ein weiteres Problem ist die Wahl von geeigneten Parametern, mit denen die Verfahren sicher genug aber gleichzeitig noch effizient in der Praxis sind. Bisherige Implementierungen sind wenig praxistauglich aus folgenden Gründen:

1. Die Implementierungen laufen nur in unrealistischen Umgebungen ohne Cloudanbindung (insbesondere keine Cloud-Datenbanken als Dienst). Beispielsweise wird die oben beschriebene delegierte Verschlüsselung zwar in [GHS12] und [CLT14] für AES und in [LN14] für ein anderes symmetrische Verfahren (namens SIMON) getestet jedoch nicht innerhalb einer realistischen Cloudumgebung. Nach der delegierten Verschlüsselung werden bei den bisherigen Tests auch keine weiteren Operationen auf den verschlüsselten Daten ausgeführt. Es fehlt daher bisher der entscheidende Nachweis, ob dieses Verfahren der delegierten Verschlüsselung in der Praxis so funktioniert, dass die Daten noch weiterverarbeitet werden können.
2. Die Implementierungen sind optimiert für eingeschränkte Spezialanwendungen (wie etwa Assoziationsstudien auf verschlüsselten Genom-Daten [KL15]). Wünschenswert wäre jedoch ein Verschlüsselungsdienst, der vielseitiger einsetzbar ist.
3. Die Implementierungen vergleichen nicht mehrere Verfahren und bieten daher keinen Anhaltspunkt, welches Verfahren am besten geeignet ist. Die einzige Ausnahmen sind hier [LN14], die zwei homomorphe Verfahren vergleichen und [CS16], die vier homomorphe Verfahren vergleichen; jedoch wurde die Implementierungen nicht in Cloudumgebungen oder einem Datenbanksystem getestet.

Für einen realistischen Einsatz von homomorphen Verschlüsselungsverfahren in Cloud-Datenbanken ergeben sich zahlreiche funktionale Anforderungen:

Nichtinteraktivität: Mehrere Kommunikationsrunden (zwischen Server und Kunden) sind für eine Datenbankanwendung zu ineffizient. Daher sollte idealerweise eine Datenabfrage jeweils nur eine einmalige Kommunikation zwischen Datenbank und Nutzer erfordern. Die homomorphen Verfahren sind grundsätzlich für diese Funktionalität geeignet. Diese Anforderung kann gegebenenfalls abgeschwächt werden: zu Gunsten einer besseren Performanz können homomorphen Verfahren mit Mehrparteien-Berechnungen kombiniert werden [LTV17].

System-Unabhängigkeit: Die bestehenden Cloud-Datenbanken sollen in ihrer Funktionsweise möglichst wenig verändert werden. Die Weiterentwicklung der Datenbanksysteme soll unabhängig von den Sicherheitsfunktionalitäten erfolgen können. Zugleich soll die Umsetzung der homomorphen Addition und Multiplikation innerhalb des Datenbanksystems erfolgen. Idealerweise sollen die Berechnungen auf verschlüsselten Daten daher durch benutzerdefinierte Funktionen in der Datenbank umgesetzt werden, die die Funktionalität des Datenbanksystems erweitern.

Modularität: Die Implementierung soll sich nicht auf ein spezielles Verschlüsselungsverfahren festlegen, sondern es sollen sich verschiedene Verschlüsselungsalgorithmen einbinden lassen. Damit wird sichergestellt, dass aktuelle Entwicklungen im Bereich der homomorphen Verschlüsselung direkt genutzt werden können.

Konfigurierbarkeit: Bei den einzelnen homomorphen Verfahren müssen konkrete Parameter gefunden werden, die eine gute Abwägung zwischen Effizienz und Sicherheit darstellen [CS16]; solche Parameter sind zum Beispiel die Schlüssellänge, der Modulus, die Länge der zu verschlüsselnden Worte oder die Anzahl der Multiplikationen. Diese Abwägung sollte die Dauer des Schutzbedarfs einschließen: Transaktionsdaten verfallen oft direkt nach der Transaktion, langzeitgespeicherte Daten brauchen stärkere Schlüssel. Dies wird auch in einem Bericht der ENISA berücksichtigt [Sm14].

Schnittstelle mit delegierter Verschlüsselung: Für die Umsetzung der delegierten Verschlüsselung muss sowohl ein geeignetes homomorphes Verfahren gefunden werden, das auf dem Clouddienst ausgeführt wird, als auch ein symmetrisches Verfahren, das auf Nutzerseite ausgeführt wird. Zusätzlich müssen Verschlüsselungs- und Entschlüsselungsmethoden des symmetrischen Verfahrens homomorph innerhalb des Datenbankservers implementiert werden können. Bei der Implementierung der delegierten Verschlüsselung ist die Wahl der Parameter nicht nur für die Sicherheit entscheidend, sondern auch für die Funktionalität: Da die Entschlüsselungsfunktion des symmetrischen Verfahrens auf Seiten der Cloud-Datenbank homomorph ausgeführt werden muss, muss das homomorphe Verfahren eine ausreichende Anzahl von Multiplikationen ausführen können.

5 Sicherheitsanforderungen

Die Sicherheit der homomorphen Verfahren beruht auf der Komplexität verschiedener theoretischer Probleme wie oben beschrieben. Jedoch werden hier oft praktische Aspekte außer Acht gelassen. Ein wichtiger Aspekt insbesondere für Datenbankanwendungen ist es, dass ein Angreifer adaptiv sein kann: aus einer Anfrage- und Antworthistorie für einen Benutzer kann er Informationen ansammeln und damit Kenntnisse über gespeicherte Daten aber auch über neu hinzugefügte Daten gewinnen. Für eigenschaftsbewahrende Kryptoverfahren wurde dieses Problem bereits erkannt [Ke15; NKW15]. Diese Art von Angriffen für homomorphe Verfahren wurden bisher nicht umfangreich untersucht.
Um einen vollen Funktionsumfang zu erreichen, muss ein Attribut gegebenenfalls mit mehreren Verfahren verschlüsselt werden; zum Beispiel sowohl ordnungsbewahrend (um sortieren zu können), durchsuchbar (um nach Suchwörtern filtern zu können) als auch homomorph (um Funktionen berechnen zu können). Dadurch liegen zu einem Klartext mehrere Schlüsseltexte und mehrere Metadaten (zum Beispiel Indexe oder Wörterbücher für ordnungsbewahrende Verschlüsselung) vor. Möglicherweise ergibt sich durch Kenntnis verschiedener Schlüsseltexte und Metadaten des gleichen Klartextes ein Informationsgewinn gegenüber der Kenntnis nur eines Schlüsseltextes. Eine offene Frage ist also, ob die einzelnen Verfahren ihre Sicherheitseigenschaften überhaupt aufrecht erhalten können, wenn mehrere Verschlüsselungen gleichzeitig verwendet werden; dies insbesondere auch in dem Fall, dass der Angreifer sich adaptiv verhält – also die Anfrage-/Antworthistorie eines Benutzers miteinbezieht.

Bei der Sicherheitsanalyse der kryptographischen Algorithmen gehen wir von einem Angreifer aus, der ehrlich aber neugierig (engl.: honest but curious) ist. Dabei beobachtet der Datenbankserver den Inhalt der verschlüsselten Datenbank, die Anfragen und Ergebnisse und versucht, daraus auf den Klartextinhalt zu schließen. Es gibt auch böswillige Angreifer, die den Datenbankinhalt oder Anfrageergebnisse aktiv manipulieren. Jedoch ist jedes homomorphe Verfahren formbar (engl.: malleable): eine Berechnung auf verschlüsselten Eingaben ergibt ein sinnvolles entschlüsselbares Ergebnis. Daher muss dem Datenbankserver in der Regel vertraut werden, dass er die Berechnungen auf den verschlüsselten Eingaben korrekt durchführt. Zusätzliche Integritätsüberprüfungen (zum Beispiel auf Grundlage von nicht-interaktiven Zero-Knowledge-Beweisen) sind möglichen, erfordern aber zusätzlichen Rechenaufwand. Alternativ gibt es erweiterte homomorphe Verfahren, die eine eingeschränkte Menge von berechenbaren Funktionen erlauben [BSW12] oder von Benutzerseite ein Token zur effizienten Berechnung auf den verschlüsselten Daten benötigen [De17].

6 Zusammenfassung

Effiziente und sichere Datenhaltung in der Cloud ist ein zukunftsrelevantes Thema. Insbesondere Cloudatenbanken spielen dabei eine große Rolle, weil sie umfangreiche Funktionalitäten zur Datenverarbeitung anbieten. Zusammenfassend lässt sich sagen, dass eine externe Speicherung von stark verschlüsselten Daten durch einen Benutzer nicht sinnvoll ist, wenn er bei jedem Lesezugriff die Daten komplett herunterladen, entschlüsseln und durchsuchen muss. Daher müssen Datenbanken, die für Cloudspeicher-Anwendungen genutzt werden, entsprechend verschlüsselte Daten (vor-)verarbeiten können.
Auf Grundlage der hier gegebenen Übersicht und Anforderungsanalyse werden wir in zukünftigen Arbeiten das bestehende System um homomorphe Verfahren erweitern. Ein wesentlicher Arbeitsschritt wird es sein, sichere und effiziente Kombinationen von eigenschaftsbewahrenden und homomorphen Verschlüsselungsverfahren zu ermitteln.

Danksagung: Tim Waage und Daniel Homann wurden zum Teil finanziert durch die Deutsche Forschungsgemeinschaft im Projekt Wi 4086/2-2.

Literatur

[ABD16] Albrecht, M. R.; Bai, S.; Ducas, L.: A Subfield Lattice Attack on Overstretched NTRU Assumptions - Cryptanalysis of Some FHE and Graded Encoding Schemes. In: CRYPTO 2016. Bd. 9814. LNCS, Springer, S. 153–178, 2016.

[Ar13] Arasu, A.; Blanas, S.; Eguro, K.; Joglekar, M.; Kaushik, R.; Kossmann, D.; Ramamurthy, R.; Upadhyaya, P.; Venkatesan, R.: Secure database-as-a-service with Cipherbase. In: SIGMOD 2013. ACM, S. 1033–1036, 2013.

[BGV14] Brakerski, Z.; Gentry, C.; Vaikuntanathan, V.: (Leveled) Fully Homomorphic Encryption without Bootstrapping. TOCT 6/3, 13:1–13:36, 2014.

[Bo13] Bos, J. W.; Lauter, K. E.; Loftus, J.; Naehrig, M.: Improved Security for a Ring-Based Fully Homomorphic Encryption Scheme. In: Cryptography and Coding 2013. Bd. 8308. LNCS, Springer, S. 45–64, 2013.

[Br] Brenner, M.; Dai, W.; Halevi, S.; Han, K.; Jalali, A.; Kim, M.; Laine, K.; Malozemoff, A.; Paillier, P.; Polyakov, Y. et al.: A standard API for RLWE-based homomorphic encryption, http://homomorphicencryption.org/white_papers/API_homomorphic_encryption_white_paper.pdf.

[Br12] Brakerski, Z.: Fully Homomorphic Encryption without Modulus Switching from Classical GapSVP. In: CRYPTO 2012. Bd. 7417. LNCS, Springer, S. 868–886, 2012.

[BS13] Brenner, M.; Smith, M.: Caching Oblivious Memory Access: An Extension to the HCRYPT Virtual Machine. In: CCS 2013. ACM, S. 1363–1366, 2013.

[BS14] Bajaj, S.; Sion, R.: TrustedDB: A Trusted Hardware-Based Database with Privacy and Data Confidentiality. IEEE Trans. Knowl. Data Eng. 26/3, S. 752–765, 2014.

[BSW12] Boneh, D.; Segev, G.; Waters, B.: Targeted malleability: homomorphic encryption for restricted computations. In: Proceedings of the 3rd Innovations in Theoretical Computer Science Conference. ACM, S. 350–366, 2012.

[Ca15] Canteaut, A.; Carpov, S.; Fontaine, C.; Lepoint, T.; Naya-Plasencia, M.; Paillier, P.; Sirdey, R.: How to Compress Homomorphic Ciphertexts. IACR Cryptology ePrint Archive 2015/113, 2015.

[CJL16] Cheon, J. H.; Jeong, J.; Lee, C.: An algorithm for NTRU problems and cryptanalysis of the GGH multilinear map without a low-level encoding of zero. LMS Journal of Computation and Mathematics 19/A, S. 255–266, 2016.

[CL15] Castagnos, G.; Laguillaumie, F.: Linearly Homomorphic Encryption from DDH. In: RSA Conference. Bd. 9048. LNCS, Springer, S. 487–505, 2015.

[CLT14] Coron, J.-S.; Lepoint, T.; Tibouchi, M.: Scale-Invariant Fully Homomorphic Encryption over the Integers. In: Public-Key Cryptography 2014. Bd. 8383. LNCS, Springer, S. 311–328, 2014.

[CS16] Costache, A.; Smart, N. P.: Which Ring Based Somewhat Homomorphic Encryption Scheme is Best? In: RSA Conference 2016. Bd. 9610. LNCS, Springer, S. 325–340, 2016.

[CS98] Cramer, R.; Shoup, V.: A Practical Public Key Cryptosystem Provably Secure Against Adaptive Chosen Ciphertext Attack. In: CRYPTO 1998. Bd. 1462. LNCS, Springer, S. 13–25, 1998.

[De10] Desmedt, Y.; Gennaro, R.; Kurosawa, K.; Shoup, V.: A New and Improved Paradigm for Hybrid Encryption Secure Against Chosen-Ciphertext Attack. J. Cryptology 23/1, S. 91–120, 2010.

[De17] Desmedt, Y.; Iovino, V.; Persiano, G.; Visconti, I.: Controlled homomorphic encryption: definition and construction. In: International Conference on Financial Cryptography and Data Security. Springer, S. 107–129, 2017.

[Di10] van Dijk, M.; Gentry, C.; Halevi, S.; Vaikuntanathan, V.: Fully Homomorphic Encryption over the Integers. In: EUROCRYPT 2010. Bd. 6110. LNCS, Springer, S. 24–43, 2010.

[FLA11] Fousse, L.; Lafourcade, P.; Alnuaimi, M.: Benaloh's Dense Probabilistic Encryption Revisited. In: AFRICACRYPT 2011. Bd. 6737. LNCS, Springer, S. 348–362, 2011.

[FV12] Fan, J.; Vercauteren, F.: Somewhat Practical Fully Homomorphic Encryption. IACR Cryptology ePrint Archive 2012/144, 2012.

[Ga84] Gamal, T. E.: A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. In: CRYPTO 1984. Bd. 196. LNCS, Springer, S. 10–18, 1984.

[Ge09] Gentry, C.: Fully homomorphic encryption using ideal lattices. In: STOC 2009. ACM, S. 169–178, 2009.

[Ge10] Gentry, C.: Computing arbitrary functions of encrypted data. Commun. ACM 53/3, S. 97–105, 2010.

[GHS12] Gentry, C.; Halevi, S.; Smart, N. P.: Homomorphic Evaluation of the AES Circuit. In: CRYPTO 2012. Bd. 7417. LNCS, Springer, S. 850–867, 2012.

[GM84] Goldwasser, S.; Micali, S.: Probabilistic Encryption. J. Comput. Syst. Sci. 28/2, S. 270–299, 1984.

[HPS98] Hoffstein, J.; Pipher, J.; Silverman, J. H.: NTRU: A ring-based public key cryptosystem. In: International Algorithmic Number Theory Symposium. Springer, S. 267–288, 1998.

[JL13] Joye, M.; Libert, B.: Efficient Cryptosystems from 2^k-th Power Residue Symbols. In: EUROCRYPT 2013. Bd. 7881. LNCS, Springer, S. 76–92, 2013.

[Ke15] Kerschbaum, F.: Frequency-hiding order-preserving encryption. In: Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security. ACM, S. 656–667, 2015.

[KF16] Kirchner, P.; Fouque, P.-A.: Comparison between Subfield and Straightforward Attacks on NTRU. IACR Cryptology ePrint Archive 2016/717, 2016.

[KL15] Kim, M.; Lauter, K.: Private genome analysis through homomorphic encryption. BMC medical informatics and decision making 15/5, S3, 2015.

[Ko87] Koblitz, N.: Elliptic curve cryptosystems. Mathematics of computation 48/177, S. 203–209, 1987.

[LN14] Lepoint, T.; Naehrig, M.: A Comparison of the Homomorphic Encryption Schemes FV and YASHE. In: AFRICACRYPT 2014. Bd. 8469. LNCS, Springer, S. 318–335, 2014.

[LTV17] López-Alt, A.; Tromer, E.; Vaikuntanathan, V.: Multikey Fully Homomorphic Encryption and Applications. SIAM Journal on Computing 46/6, S. 1827–1892, 2017.

[NKW15] Naveed, M.; Kamara, S.; Wright, C. V.: Inference attacks on property-preserving encrypted databases. In: Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security. ACM, S. 644–655, 2015.

[NLV11] Naehrig, M.; Lauter, K. E.; Vaikuntanathan, V.: Can homomorphic encryption be practical? In: Cloud Computing Security Workshop (CCSW) 2011. ACM, S. 113–124, 2011.

[Pa99] Paillier, P.: Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. In: EUROCRYPT 1999. Bd. 1592. LNCS, Springer, S. 223–238, 1999.

[PBS11] Perl, H.; Brenner, M.; Smith, M.: Poster: An Implementation of the Fully Homomorphic Smart-Vercauteren Cryptosystem. In: CCS 2011. ACM, S. 837–840, 2011.

[Po12] Popa, R. A.; Redfield, C. M. S.; Zeldovich, N.; Balakrishnan, H.: CryptDB: processing queries on an encrypted database. Commun. ACM 55/9, S. 103–111, 2012.

[PZB15] Popa, R. A.; Zeldovich, N.; Balakrishnan, H.: Guidelines for Using the CryptDB System Securely. IACR Cryptology ePrint Archive 2015/979, 2015.

[RAD78] Rivest, R. L.; Adleman, L.; Dertouzos, M. L.: On data banks and privacy homomorphisms. Foundations of secure computation 4/11, S. 169–180, 1978.

[Sh17] Shameer, K.; Badgeley, M. A.; Miotto, R.; Glicksberg, B. S.; Morgan, J. W.; Dudley, J. T.: Translational bioinformatics in the era of real-time biomedical, health care and wellness data streams. Briefings in bioinformatics 18/1, S. 105–124, 2017.

[Sm14] Smart, N. P.; Rijmen, V.; Gierlichs, B.; Paterson, K.; Stam, M.; Warinschi, B.; Watson, G.: Algorithms, key size and parameters report, Techn. Ber., European Union Agency for Network und Information Security, 2014.

[Tu13] Tu, S.; Kaashoek, M. F.; Madden, S.; Zeldovich, N.: Processing Analytical Queries over Encrypted Data. VLDB 6/5, S. 289–300, 2013.

[Wi15] Wiese, L.: Advanced Data Management: For SQL, NoSQL, Cloud and Distributed Databases. Walter de Gruyter GmbH & Co KG, 2015.

[WW17] Waage, T.; Wiese, L.: Property Preserving Encryption in NoSQL Wide Column Stores. In: Cloud and Trusted Computing 2017. Bd. 10574. LNCS, Springer, S. 3–21, 2017.