Benutzerfreundliche Verschlüsselung für Cloud-Datenbanken

Lena Wiese · Tim Waage
Forschungsgruppe Knowledge Engineering
Institut für Informatik
Georg-August-Universität Göttingen
{lena.wiese | tim.waage}@cs.uni-goettingen.de

Zusammenfassung

In den letzten Jahren erfreuen sich sogenannte Cloud-Speicherdienste wachsender Beliebtheit, in deren Hintergrund die Informationen in verteilten Datenbanken abgelegt werden. Populäre Vertreter sind NoSQL-Datenbanken, die durch ihre exzellenten Eigenschaften hinsichtlich Skalierbarkeit und Ausfallsicherheit sehr gut für dieses Szenario geeignet sind. Unglücklicherweise bieten diese aber nicht die gewohnten Sicherheitsmechanismen etablierter SQL-Plattformen. Um dem entgegenzuwirken ohne die eigentliche Implementation der Datenbanken ändern zu müssen, bietet es sich an die abzulegenden Daten zu verschlüsseln. Naturgemäß schränkt dies aber die Möglichkeiten ein mit den Daten zu interagieren, beispielsweise sie zu durchsuchen oder nach bestimmten Kriterien zu sortieren. Unser Lösungsansatz ist daher ein Proxy-Client zwischen der eigentlichen Applikation und der Datenbank, der alle ausgehenden Daten nach diversen Schemata zur durchsuchbaren und ordnungserhaltenden so verschlüsselt, dass die Funktionalität der Datenbanken nicht eingeschränkt wird. Eine Herausforderung sind die sehr unterschiedlichen Schnittstellen und Datenmodelle der einzelnen NoSQL Datenbanken. Wir haben zwei Schemata zur durchsuchbaren Verschlüsselung (sowohl scan- als auch index-basiert) für die populären NoSQL-Datenbanken Apache Cassandra und Apache HBase implementiert, präsentieren praktische Optimierungsansätze und zeigen, dass der dadurch verursachte Geschwindigkeitsverlust in Anbetracht zur gewonnenen Sicherheit gut vertretbar ist.

1 Einleitung

Einhergehend mit der starken Zunahme der Menge zu speichernder Daten wurden in letzter Zeit Datenmodelle und Datenbanksysteme entwickelt, die von der klassischen relationalen (also tabellenorientierten) Sichtweise auf die Daten abweichen. Eine Variante dieser modernen Datenbanksysteme sind die sogenannten Spaltenfamilien-Datenbanken. Es existieren derzeit zahlreiche sowohl proprietäre als auch quelloffene Implementierungen (wie Apache Cassandra oder Apache HBase), die in verschiedenen Bereichen Anwendung finden. Ein Anwendungsbereich ist der Einsatz als Cloud-Datenbank, wobei sich Kunden Speicherplatz bei einem externen Anbieter (wie etwa Amazon) mieten können. Cloud-Datenbanken können enorme Mengen an Daten speichern und bereitstellen. Die Mehrheit der verfügbaren Systeme bieten jedoch keine nativen Sicherheitsmechanismen. Es wird im Allgemeinen angenommen, dass das beispielsweise das Frontend des Datenbankanbieters Zugriffskontrolle, Rechtevergabe und andere derartige Funktionalitäten bereitstellt. Eine Möglichkeit sensible Daten zu schützen, ohne sich hierauf verlassen zu müssen, ist der Einsatz von Verschlüsselung. Derzeit unterstützt jedoch keine der vorhandenen Implementierungen eine verschlüsselte Speicherung von Daten – insbesondere keine durch den Benutzer konfigurierbare Verschlüsselung.

P. Schartner · K. Lemke-Rust · M. Ullmann (Hrsg.) · D•A•CH Security 2015 · syssec (2015) 12-23.

Unsere bisherigen Arbeiten setzen kryptographische Sicherheitsfunktionen für Spaltenfamilien-Datenbanken um und analysieren sie eingehend – idealerweise ohne die generelle Funktionsweise (und damit die bestehenden Implementierungen) von Spaltenfamilien-Datenbanken ändern zu müssen.

2 Die Plattform

Der Lösungsansatz innerhalb des Projekts besteht darin, eine zusätzliche Software (einen Proxy-Client) zu entwickeln, die auf dem Rechner des Benutzers ausgeführt wird und dort die Schlüsselverwaltung sowie die Ver- und Entschlüsselung übernimmt; diese Software wird also zwischen den Benutzer (beziehungsweise die Zugriffsschnittstelle der Datenbank) und die Clouddatenbank zwischengeschaltet.

Der Proxy-Client hat dabei idealerweise folgende grundlegende Eigenschaften:

- Datensicherheit: Grundsätzlich sollen nur verschlüsselte Daten den Rechner des Benutzers verlassen. Dabei müssen jedoch Verschlüsselungsverfahren eingesetzt werden, die ein Datenmanagement durch die Clouddatenbank ermöglichen; der Stand der Technik solcher Verfahren wird im folgenden Abschnitt kurz dargestellt.
- Benutzerfreundlichkeit: Ein hoher Grad an Usability wird erreicht durch eine benutzerfreundliche Schnittstelle mit verständlichen Erläuterungen zur Vorgehensweise der Software, die interaktive während der Benutzung der Software angezeigt werden und die dem Benutzer eine selbstbestimmte Konfiguration der Verschlüsselungssoftware ermöglichen.
- Plattformunabhängigkeit: Durch Anbindung verschiedener Clouddatenbank-Anbieter wird eine starke Bindung an eine Plattform (engl.: vendor lock-in) vermieden. Die einfache Übertragbarkeit der Daten auf einen anderen Anbieter wird dadurch möglich.

Durch diese Eigenschaften eignet sich der Prototyp dieses Projektes langfristig gesehen als sichere Speicherlösung für verschiedene Anwendungsfälle, wie zum Beispiel:

- Selbstbestimmte, selektive Herausgabe von persönlichen Informationen: In sozialen Netzwerken oder auf Karriereplattformen kann der Benutzer selbst entscheiden, welche Inhalte er verschlüsselt und durch geeignetes Schlüsselmanagement nur einem eingeschränkten Benutzerkreis zugänglich macht.
- Erschwerung von Profilbildung: Durch die eingesetzten Verschlüsselungsalgorithmen werden analytische Anfragen (etwa zur Erstellung von Profilen einzelner Nutzer) nur bis zu einem gewissen Grad unterstützt. Dies führt zu einer erhöhten Privatsphäre gegenüber Empfehlungssystemen (z.B. eBay, Amazon, Netflix) und gegenüber personalisierter Werbung.
- Verschlüsselte Nachrichtendienste: E-Mail-Dienste, Webseiten mit Benachrichtigungsfunktionen (z.B. Facebook), Mailverteiler (z.B. Yahoo Groups) und Web-Foren können verschlüsselte Nachrichten verwalten und dabei dennoch Textsuchen und Sortierung unterstützen. Der Dienstanbieter kann aber keine vertraulichen Nachrichteninhalte mehr ausspähen (wie persönliche Daten oder private Verabredungen).
- Schutz von Kundendaten vor Hackerangriffen: Durch Hackerangriffe sind in letzter Zeit Millionen von Kundendaten kompromittiert worden (so etwa beim Sony-Hack von 2011 [Spi]). Durch eine weitgehend verschlüsselte Speicherung von Kundendaten, können die Auswirkungen eines solchen Datendiebstahls beschränkt werden. Dies gilt insbesondere

für sehr private Daten wie zum Beispiel in digitalen Krankenakten.

Eine besondere Schwierigkeit besteht darin, dass je nach Clouddatenbank unterschiedliche Zugriffsmethoden und unterschiedliche Datenmodelle benutzt werden. Um einen möglichst breiten Einsatz in der Praxis bieten zu können, werden deshalb die populären NoSQL Datenbanken Apache Cassandra und Apache HBase unterstützt. Beide sind weit verbreitet, basieren in den Grundzügen ihrer Datenmodelle auf sehr ähnlichen Konzepten und werden aktiv weiterentwickelt, verfolgen aber jeweils vollkommen unterschiedliche architektonische Konzepte.

- Apache Cassandra bietet mit der Cassandra Query Language (CQL) eine Abfragesprache, die sehr stark an SQL angelehnt ist. Aufgrund der Schemafreiheit des zugrundeliegenden Datenmodells ist sie jedoch nicht ganz so umfangreich, aber für mehrlagige Verschlüsselung ähnlich wie in [PRZB12] sehr gut geeignet. Es existieren des Weiteren zahlreiche Treiber, die CQL für diverse verbreitete Programmiersprachen wie Python, C#, .NET, Java, Ruby u.a. nutzbar macht. Innerhalb des Projektes wird hauptsächlich in Java entwickelt, wofür entsprechende Treiber zur Verfügung stehen.
- Apache HBase, basierend auf Apache Hadoop, verfolgt einen anderen Ansatz und ist in erster Linie mehr als Data Store zu verstehen, denn es bringt weniger Funktionalität mit, als man es von klassischen relationalen Datenbanken und auch einigen anderen NoSQL-Datenbanken gewohnt ist. Es gibt im Wesentlichen vier Operationstypen (get, put, scan und delete) zur Interaktion mit der Datenbank, die dafür sehr gut geeignet ist um mit großen Datenmengen mit Millionen oder sogar Milliarden Datensätzen umzugehen. HBase bringt eine native Java API mit, die im Projekt verwendet wird.

3 Verschlüsselungsverfahren

Es existieren starke kryptographische Verfahren, um den Schutz vertraulicher Daten beweisbar sicherzustellen. Neben dem Vertraulichkeitsschutz gilt es aber auch die Daten effizient verarbeiten zu können. Der Schutz der Daten und die Nutzbarkeit der Daten sind daher in der Regel zwei gegensätzliche Anforderungen. Idealerweise sollte eine (Vor-)Verarbeitung auf verschlüsselte Daten möglich sein, um eine vollständige Entschlüsselung, Verarbeitung und anschließende Neu-Verschlüsselung zu vermeiden. 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.

Für Anwendungen im Datenbankbereich bieten sich hier folgende Verfahren an:

- Durchsuchbare Verschlüsselung: Abstrakt betrachtet erstellt ein Verfahren zur durchsuchbaren Verschlüsselung einen Suchindex für eine Menge von Daten. Der Suchindex enthält vorher zu definierenden Schlüsselworte und die Indexeinträge verweisen dann auf die Datensätze, die das jeweilige Schlüsselwort enthalten. Der Index wird derart verschlüsselt, dass nur mit Hilfe einer Falltür (engl.: trapdoor) ein Vergleich von einem Suchwort mit den im Index enthaltenden Schlüsselwörtern möglich ist. Hierbei gibt es symmetrische Verfahren (engl.: searchable symmetric encryption, kurz SSE) und asymmetrische Verfahren (engl.: public key encryption with keyword search, kurz PEKS). Bei einem symmetrischen Verfahren gibt es einen geheimen Schlüssel, der sowohl für die Verschlüsselung des Indexes als auch für die Suche verwendet wird; als grundlegendes

Verfahren gilt [SWP00]. Bei einem asymmetrischen Verfahren wird die Verschlüsselung mit dem öffentlichen Schlüssel vorgenommen und eine Suche ist nur mit einer Falltür möglich, die in Abhängigkeit von dem passenden privaten Schlüssel generiert wurde; als grundlegendes Verfahren gilt [BDCOP04]. Eine umfassende Übersicht über die derzeitigen Erweiterungen, die aus den grundlegenden Verfahren entwickelt wurden, liefern [ABC+05, Tan13]. Für durchsuchbare Verschlüsselung gibt es zudem Sicherheitsgarantien gegen verschiedene Angriffsmodelle (zum Beispiel chosen keyword attacks, CKA), die ebenfalls in [Tan13] beschrieben werden. Eine weitere relevante Arbeit ist die von Kamara, Papamanthou und Roeder [KPR12], die zum Einen eine Übersicht über bisher entwickelte SSE-Verfahren und deren Sicherheitsgarantien gibt, zum Anderen dynamisches SSE einführt, mit dem Änderungen an den verschlüsselten Daten effizient möglich wird.

- Ordnungsbewahrende Verschlüsselung: Mit ordnungsbewahrender Verschlüsselung für numerische Daten überträgt sich eine Ordnungsrelation von den Klartexten auf die Schlüsseltexte: wenn für zwei Klartexte a und b gilt, dass $a < b$, dann gilt für die Verschlüsselungen $Enc(a) < Enc(b)$. Das grundlegende Verfahren wurde in [AKSX04] entwickelt und [BCO11] liefern aktuelle Erkenntnisse. [Tan10] beschreibt ein verwandtes Verfahren, das Vergleiche auf Schlüsseltexten ermöglicht. Eine grundlegende Analyse von Sicherheitseigenschaften ordnungsbewahrender Verschlüsselung unter Angriffen mit bekannten Klartexten (engl.: known plaintext attacks) wird in [XY12] durchgeführt. Besonders hervorzuheben sind dabei die Verfahren [G+03, RVBM09, WWS05, DCdVFJ+11], die Bloomfilter und Indexe über verschlüsselten Daten verwenden.
- Homomorphe Verschlüsselung: Homomorphe Verschlüsselung [VDGHV10] erlaubt es, Berechnungen auf verschlüsselten Daten auszuführen, so dass verschlüsselte Ergebnisse herauskommen. Generelle homomorphe Verfahren sind jedoch extrem ineffizient und führen zu hohen Laufzeitverlusten. Daher ist es für eine praktische Anwendung notwendig, das Verschlüsselungsverfahren für die spezifische Anwendung zu optimieren bzw. sich direkt für ein homomorphes Verfahren zu entscheiden, welches zumindest die benötigte Funktionalität effizient abwickelt. So ist es etwa ausreichend für die Berechnung von Summen, additive Homomorphismen zu benutzen [TKMZ13, Pai99].

4 Ergebnisse

Verschlüsselung geht mit dem Verlust von Performanz einher. Bisherige Ergebnisse quantifizieren diesen Verlust bei Nutzung konkret existierender Verschlüsselungsverfahren und Datenbanksysteme. Zum Einsatz kommen – wie oben beschrieben – Apache Cassandra und Apache HBase, zwei populäre Vertreter aus dem Bereich der NoSQL Datenbanken, die momentan zunehmend Beachtung bei Cloud-Service-Anbietern finden.

4.1 Symmetrische Verschlüsselung

Als Grundlage für die Performanzmessungen dient der Yahoo! Cloud Serving Benchmark (YSCB), ein Java-Framework für die Erfassung der Leistung von Key-Value-Datenbanken, zu denen auch Cassandra und HBase gehören. Erfasst werden die Auswirkungen verschiedener Parameter auf den Gesamtdurchsatz des Datenbanksystems, sowie auf Lese- und Schreiblatenz. Im Betrieb ohne Verschlüsselung liest und schreibt der YCSB eine Reihe von zufällig generierten Schlüssel-Wert-Paaren in die Zieldatenbank und liest sie wieder aus. Diese Ar-

beitsweise wurde von uns um entsprechende Ver- und Entschlüsselungsschritte ergänzt, so dass die Datenbanken zu jeder Zeit nur noch verschlüsselte Daten speichern, ganz wie es im Sinne des Schutzes vertrauenswürdiger Daten notwendig ist. Die Verschlüsselung selbst wird dabei mit der AES-Implementatierubg des „Bouncy Castle"-Kryptoproviders [API] realisiert. In einem ersten Test wird der Einfluss der Anzahl der sogenannten „Worker Threads" untersucht. Dies simuliert das Datenbankverhalten mit einer zunehmenden Zahl gleichzeitig agierender Benutzer. Sowohl bei Cassandra, als auch bei HBase, lässt sich kein signifikanter damit zusammenhängender Performanzverlust feststellen. In einem zweiten Test wird der Einfluss eines gewünschten Zieldurchsatzes untersucht. Der YCSB bietet hier Möglichkeiten, die Anzahl der Operationen pro Sekunde festzulegen, um den Einfluss auf Lese- und Schreiblatenzen zu testen. Wie zu erwarten sind beide Datenbanken mit eingeschalteter Verschlüsselung deutlich früher „gesättigt" und werden nicht mehr schneller. Die Latenzen steigen ebenso bei Lese- und Schreiboperationen, nicht jedoch bei Range-Queries. Ein dritter Test untersucht die Performanz bei insgesamt steigendem Aufkommen von Datenbankoperationen, ohne jedoch wie beim vorherigen Test die Anzahl der Operationen pro Sekunde zu regulieren. Der Durchsatz der Datenbanken erhöht sich hier erwartungsgemäß, die Lese-/Schreiblatenzen werden kleiner, selbst bei aktivierter Verschlüsselung. Insgesamt stellten wir fest, dass der Einsatz von Verschlüsselung den Gesamtdurchsatz bei einzelnen Lese-/Schreiboperationen auf ca. 40% reduziert, die Lese- und Schreiblatenzen steigen um das 2 bis 3 fache. Bei Range-Queries ist die Situation deutlich schlechter. Der Durchsatz sinkt auf 10%. Leselatenzen steigen um das 10 fache, Schreiblatenzen um das 4 fache. Die Schlüssellänge hat dabei keinen signifikanten Einfluss. Während der YCSB als synthetischer Benchmark damit ein durchwachsenes Bild zeigt, sind im Weiteren natürlich Ergebnisse von Interesse, die sich auf praktischer nutzbare Verschlüsselungsverfahren beziehen.

4.2 Durchsuchbare Verschlüsselung

Die in Abschnitt 4.1 angewandte AES-Verschlüsselung begrenzt die naturgegebene Funktion einer Datenbank immens. Da diese den Klartext nicht mehr kennt, ist es beispielsweise unmöglich noch nach Dokumenten mit bestimmen Worten zu suchen. Es gibt jedoch Schemata, die Klartexte durchsuchbar verschlüsseln. Wir implementierten ein solches Schema [SWP00] (nachfolgend „SWP Schema") für die Nutzung mit Cassandra und HBase, welches die IND-CPA sichere und dennoch durchsuchbare Verschlüsselung erlaubt. Dabei werden bis zu 100.000 Worte pro Sekunde bei der Verschlüsselung und über 250.000 Worte pro Sekunde beim Durchsuchen der verschlüsselten Daten auf durchschnittlicher Notebook Hardware erreicht. Die sind Werte, die einen praktischen Nutzen dieser Technologie sinnvoll erscheinen lassen – zum Beispiel für einen praktischen Anwendungsfall: Das Verschlüsseln der privaten Mailbox. Sowohl mit Cassandra, als auch mit HBase als darunterliegendem Datenbanksystem ergeben sich brauchbare Geschwindigkeiten. So beträgt beispielsweise die durchschnittliche Dauer für das Hinzufügen einer Mail gerade einmal 0.04 Sekunden.

Im Anschluss implementierten wir noch einmal eine deutliche Erweiterung dieser Erkenntnisse. Neben dem SWP Schema, das ohne Indizes auskommt, wird hier auch eines der neuesten index-basierten Schemata, das SUISE Schema [HK14] untersucht. Ein Testdatensatz mit ca. 7 Millionen Worten in Form von 10.000 E-Mails stellt dabei erneut sicher, dass die Ergebnisse praktische Relevanz haben. Abbildung 1 und 2 zeigen die Ergebnisse.

Da beide Verfahren beim Verschlüsseln einmal über den kompletten Datensatz iterieren müssen, steigt die dafür benötigte Zeit linear mit der Größe desselben. Die verbesserte Implementatie-

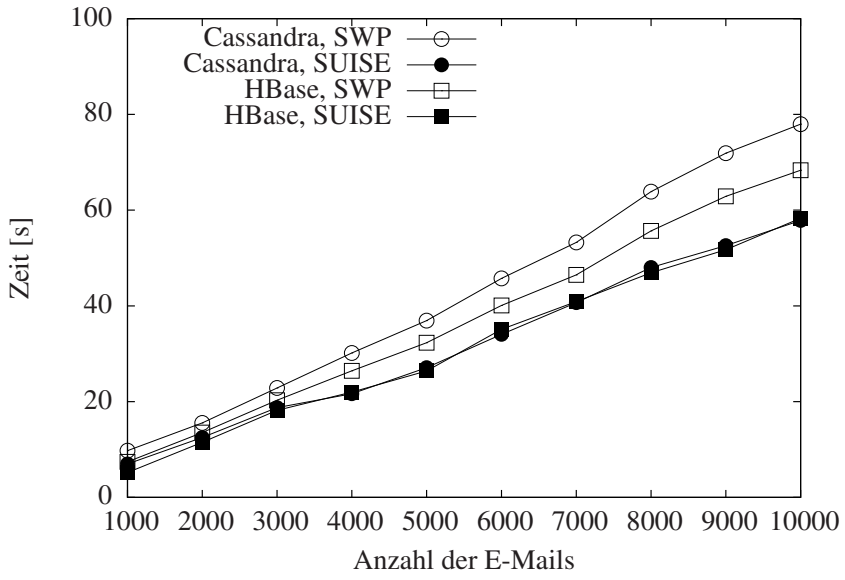

Abb. 1: Zur Verschlüsselung benötigte Zeit mit zunehmender Datensatzgröße

rung des SWP Schemas erreicht ca. 120.000 Worte pro Sekunde beim Verschlüsseln und ca. 530.000 Worte pro Sekunde beim Durchsuchen. SUISE bewegt sich auf ähnlichem Niveau mit 90.000 bzw. 530.000 Worten pro Sekunde und bietet darüber hinaus mit Hilfe eines speziellen zweiten Indexes konstante Zeiten von unter 1ms bei der Suche nach bereits zuvor verwendeten Suchworten. Zwei Dinge sind an dieser Stelle zu betonen. Einerseits mag es zunächst überraschen, dass SUISE ähnlich schnell verschlüsselt wie SWP, obwohl letzteres gar keinen Index erzeugen muss. SUISE kompensiert dies, indem es gleiche Worte pro Dokument nur einmal für die Indexerstellung berücksichtigen muss, SWP hingegen jedes Wort auch wiederholt verschlüsselt. Andererseits wurde das SWP Verfahren für diesen Vergleich insofern von uns modifiziert, dass es nach dem ersten Treffer in einem Dokument abbrechen darf. Damit liefert es die gleiche Information wie SUISE (nämlich ob ein Suchwort in einem Dokument vorkommt, oder nicht). Die Fähigkeit des SWP Algorithmus im Gegensatz zu SUISE eigentlich auch Anzahl und Position von Treffern pro Dokument zu liefern, würde ihm in diesem Vergleich sonst zum Nachteil werden.

Weiterhin werden die Optimierungspotenziale beider Schemata ausführlich beleuchtet. Anhand bestimmter Parameter kann deren Leistung für spezifische Anwendungsfälle weiter verbessert werden.

Ein Merkmal des SWP Verfahrens ist die Verschlüsselung aller Worte mit derselben Länge n. Das bedeutet, Worte des ursprünglich Klartextes, die kürzer sind als n Zeichen, werden mittels Padding (wir nutzten PKCS7 [Kal98]) verlängert. Längere Worte werden entsprechend in mehrere Fragmente der Länge n zerlegt (wobei falls nötig das letzte Fragment wieder gepaddet wird). Von entscheidender Bedeutung ist nun, wie groß n konkret gewählt wird. Padding führt zu größeren Schlüsseltexten, das Aufspalten der Worte in Fragmente wiederum zu mehr Iterationen des SWP Verfahrens. Das heißt in der Praxis: Ein zu klein gewähltes n kann einen Performanceverlust bedeuten, aber Speicherplatz sparen. Große n hingegen können das Verfahren beschleunigen, da für weniger Worte entsprechend weniger Iterationen benötigt werden, wobei aber auch der Speicherplatzbedarf steigt.

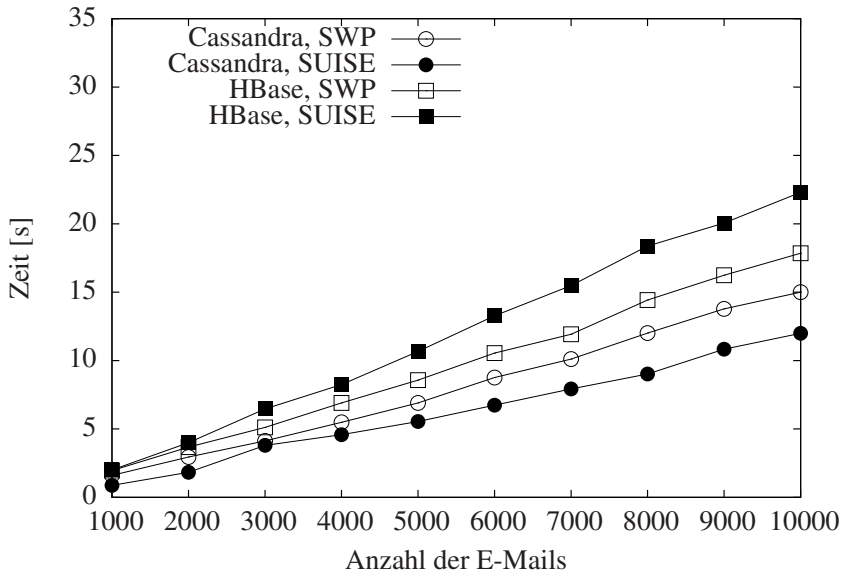

Abb. 2: Zur Suche benötigte Zeit mit zunehmender Datensatzgröße

Abbildung 3 zeigt das Ergebnis für $4 \leq n \leq 9$. Ab $n = 8$ steigt die Performance nicht mehr nennenswert an, während die Größe des Schlüsseltexts erwartungsgemäß linear zunimmt. $n = 8$ ist somit für den untersuchten Testdatensatz der optimale Kompromiss zwischen Geschwindigkeit und Speicherplatzbedarf. Dies ist natürlich kein allgemein gültiger Wert sondern muss für andere Datensätze individuell ermittelt werden. Dies kann mitunter schwierig werden, wenn diese mit der Zeit wachsen und ihre Charakteristik (insbesondere die durchschnittliche Wortlänge) ändern.

Auch das SUISE Verfahren bietet Raum für Optimierungen. Sein größter Nachteil ist der potenziell sehr große Index, der für jedes Dokument verschlüsselte Repräsentationen der darin enthaltenen Worte speichert. Eine solche Repräsentation setzt sich zusammen aus dem Output eines sog. Zufallsorakels und eines Zufallsgenerators. Für eine detaillierte Beschreibung sei an dieser Stelle auf die Originalarbeit [HK14] verwiesen. Den Empfehlungen der Autoren folgend schlägt jedes Wort im Index letztendlich mit 40 Bytes zu Buche, selbst wenn es im Klartext nur ein Byte benötigt. Der Index kann somit im Vergleich zum Klartext sehr groß werden. Man beachte, dass die verschlüsselten Daten an sich hier noch nicht einmal berücksichtigt sind. Es ist möglich den Index zu verkleinern, ohne das Verfahren unsicherer zu machen, indem die Outputlänge des Zufallsgenerators verändert wird. Die Autoren gehen hier von 160 Bits aus.

Abbildung 4 zeigt die entsprechenden Resultate. Der Speicherplatzbedarf in den Datenbanken wächst mit der Outputlänge der Zufallsgenerators. Während dies bei Cassandra noch eine signifikante Verbesserung zur Folge hat, so fällt bei HBase die Verbesserung anteilig sehr viel geringer aus.

4.3 Ordnungsbewahrende Verschlüsselung

Die ordnungsbewahrende Verschlüsselung spielt für die Funktionsweise von NoSQL-Datenbanken wie Cassandra und HBase eine besondere Rolle. Zum einen sollte sie verwendet werden, um Zeilenidentifikatioren (Rowkeys) zu verschlüsseln, da diese von der Datenbank zur Sortierung benutzt werden. Damit werden Zeilen mit ähnlichen Identifikatoren auch physisch nah

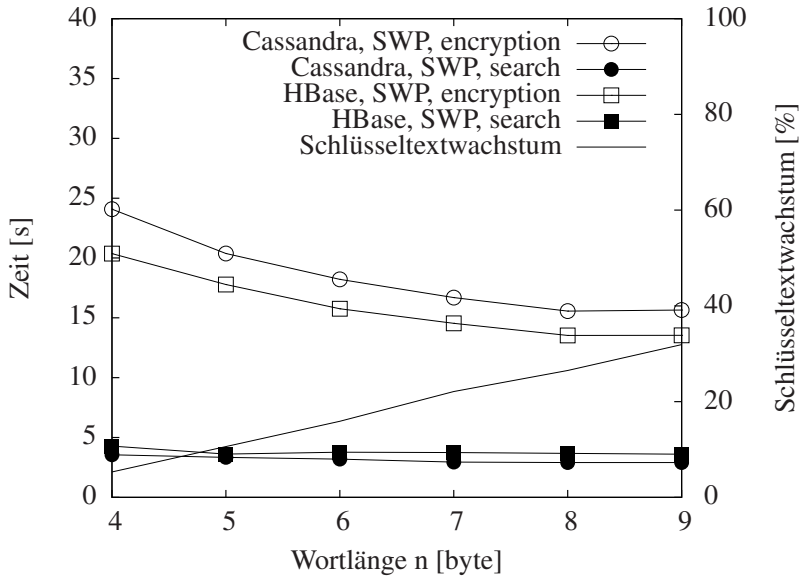

Abb. 3: Performance und Speicherplatzbedarf des SWP Verfahrens mit wachsendem n

beieinander abgelegt, was insbesondere Bereichsabfragen extrem beschleunigt. Mit den Schemata zur durchsuchbaren Verschlüsselung würde man hier zu jeweils zufällig anmutenden Identifikatoren gelangen, was die Lokalität der Daten zerstören würde mit der Konsequenz des entsprechenden Performance- und Funktionalitätsverlusts. Zum anderen sollte ordnungsbewahrende Verschlüsselung bei allen Informationen angewandt werden, die für die Datenbank relevant für Zeitangaben (und somit Versionierungen) sind. Das betrifft in allererster Linie natürlich Timestamps, die auch nach der Verschlüsselung noch Auskunft darüber geben können müssen, welcher Datensatz/Version älter oder jünger ist. Auch die Funktionalität der Löschmarkierungen (Tombstones) ist davon betroffen.

Vergleichbar zur Menge der Schemata zur durchsuchbaren Verschlüsselung gibt es auch zur ordnungsbewahrenden Verschlüsselung diverse Ansätze, die sich im Wesentlichen aber in zwei Kategorien aufteilen ließen, nämlich ob das Schema zur Ver- und Entschlüsselung Statusinformationen benötigt (zustandsorientiert) oder nicht (nicht zustandsorientiert). Wir entschieden uns zur näheren Untersuchung für jeweils ein Schema.

Es stellte sich heraus, dass bei zustandsorientierten Verfahren der ordnungsbewahrenden Verschlüsselung fast immer eine Form von Wörterbuch als Zustandsspeicher genutzt wird, in der zu jeder benötigten Zahl aus einem Startbereich (die sog. Domain) die entsprechende Abbildung in einen Zielbereich (die sog. Range) vermerkt wird, so auch beim von uns näher betrachteten und implementierten Verfahren von Kerschbaum et. al [KS14]. Naturgemäß haben Ansätze dieser Art häufig das Problem, dass deren Wörterbuch neu erstellt (und damit potenziell die komplette Datenbank neu verschlüsselt) werden muss, wenn der Fall eintritt, dass in einem bestimmten Bereich der Range kein Platz mehr vorhanden ist, d.h. eine weitere Zahl zwischen zwei bereits direkt aufeinander folgenden Zahlen benötigt werden würde. Glücklicherweise kann die Range dank großer Wertebereiche für Zahlentypen in den entsprechenden Programmiersprachen und Datenbanken aber so groß gewählt werden, dass dieser Fall in der Praxis verschwindend selten tatsächlich auftritt, was unsere Implementation bestätigen konnte. Mehrere Millionen Zahlen (sowohl zufällig erzeugt, als auch aus Testdatensätzen bestehend) konnten verschlüsselt werden,

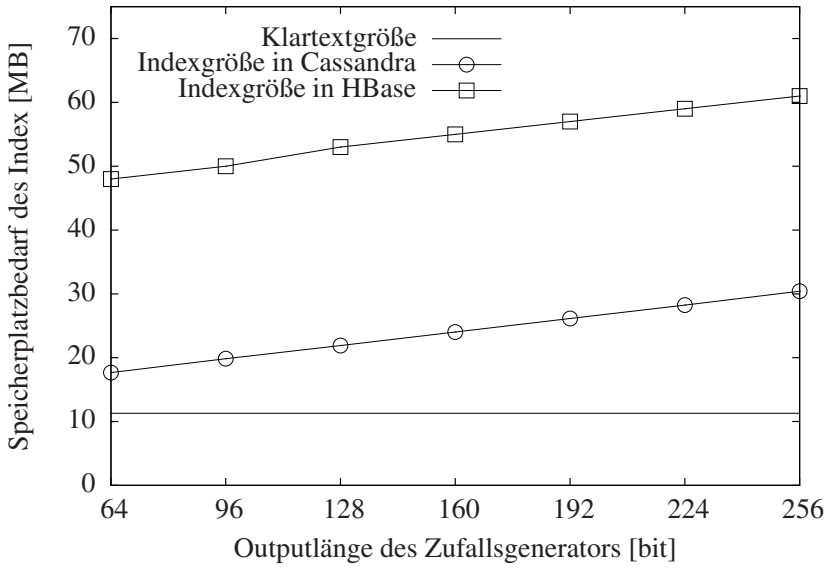

Abb. 4: Indexgröße mit wachsender Outputlänge des Zufallsgenerators in SUISE

ohne dass je eine Neuordnung notwendig gewesen wäre. Zahlenfolgen bestimmter Charakteristik sollten dennoch vermieden werden. Das worst case Szenario ist eine sortierte Zahlenfolge der Form 1, 2, ... n bzw. n, n-1, ..., 1. Damit tritt der oben beschriebene Fall schnellst möglich ein.

Nicht zustandsorientierte Schemata haben dieses Problem nicht. Wir wählten den Ansatz von Boldyreva et. al [BCO11], der auf zwei Schlüsselideen basiert. Zum einen wird das Wörterbuch dynamisch nur für den Bereich erzeugt, der gerade gebraucht wird. Zum anderen erfolgt die Auswahl einer Zahl aus der Range zu einer Zahl aus der Domain anhand der hypergeometrischen Zufallsverteilung. Das hat im Gegensatz zum Ansatz von Kerschbaum et. al aber den Nachteil, dass bestimmte Zahlen aus dem Schlüsseltext wahrscheinlicher sind für bestimmte Zahlen aus dem Klartext als andere, was theoretisch statistische Analysen des Schlüsseltexts ermöglicht.

Die Sicherheitseigenschaften ordnungsbewahrender Verschlüsselungen können nicht genauso betrachtet werden wie die zur durchsuchbaren Verschlüsselung. Bestimmte Informationen, nämlich die Relationen beliebiger ursprünglicher Klartextpaare, sollen gerade absichtlich erkennbar bleiben. IND-CPA-Sicherheit kann somit schon per Definition nicht erreicht werden. Stattdessen muss eine Abschwächung des IND-CPA-Begriffs vorgenommen werden zu IND-OCPA (indistinguishability under ordered chosen-plaintext), d.h. über die Klartexte wird abgesehen von ihren Relationen untereinander keinerlei weitere Information offenbart. IND-OCPA Sicherheit kann nur erreicht werden, wenn die Range exponentiell größer ist als die Domain, was wie oben beschrieben aber für nahezu alle reellen Datensätze praktisch problemlos machbar ist. Beide der gewählten Schemata zur ordnungsbewahrenden Verschlüsselung sind unter dieser Annahme IND-OCPA sicher.

[BCO11] führen darüber hinaus einen weiteren Sicherheitsbegriff für ihr Schema ein: POPF-CCA (pseudorandom order-preserving function against chosen-ciphertext attack). Damit ist gemeint, dass Schlüsseltexte der Range gleichmäßig, aber trotzdem zufällig (also ideal) den

Klartexten der Domain zugeordnet werden. Obwohl dies strenggenommen mit maschineller Berechnung nicht möglich ist, so ist ein Schema zur ordnungsbewahrenden Verschlüsselung POPF-CCA sicher, wenn es zumindest nicht berechenbar unterscheidbar ist von einer solch idealen Zuordnung.

5 Zusammenfassung und Ausblick

Das Hauptziel des beschriebenen Projektes ist es, Daten in verschlüsselter Form bei verschiedenen Clouddatenbank-Anbietern speichern zu können. Dabei sollen die Daten sowohl vor „Angriffen“ durch einen ehrlichen aber neugierigen (engl.: honest-but-curious) Cloudspeicher-Anbieter aber auch vor möglicherweise böswilligen Angriffen Dritter geschützt werden. Das Modell eines ehrlichen aber neugierigen Cloudspeicher-Anbieters bedeutet, dass der Cloudspeicher-Anbieter versucht, durch die von Kunden gesendeten Daten und Anfragen beziehungsweise durch die berechneten Antworten vertrauliche Informationen zu sammeln; der Anbieter verhält sich aber ansonsten regelkonform (er versucht also nicht böswillig Daten zu verändern oder falsche Antworten zu senden). Gleichzeitig gilt es als Nebenbedingung, das technische Vorgehen der Clouddatenbanken zu bewahren. Das bedeutet unter anderem, dass die grundlegenden Funktionalitäten (zum Beispiel Sortierung von Daten) der Datenbanksysteme und die effiziente Anfragebearbeitung durch den Datenbankserver weiterhin unterstützt werden sollen. Die Nutzbarkeit der verschlüsselten Daten für eine Verwaltung und Anfragebearbeitung durch Clouddatenbanken muss also gewährleistet bleiben. Dabei ist erkennbar, dass das Hauptziel (der Schutz vertraulicher Daten) und die Nebenbedingung (die Nutzbarkeit verschlüsselter Daten) zwei konträre Anforderungen sind. Da der Wunsch nach einer Datenverwaltung und Anfragebearbeitung durch den Cloudspeicher-Anbieter vorhanden ist, muss hier eine sinnvolle Abwägung zwischen den beiden Anforderungen gefunden werden: möglichst hohe Sicherheitsgarantien bei gleichzeitig möglichst effizienter Datenverwaltung in der Cloud. Die bisher getesteten Verschlüsselungsverfahren zeigen positive Performanzeigenschaften. In Zukunft wird die Plattform um weitere Verfahren ergänzt. Als ergänzende Analyse ist es langfristig wichtig, die Sicherheitseigenschaften der eingesetzten Verfahren konkret abschätzen zu können.

Danksagung

Das Projekt FamilyGuard wird von der DFG gefördert. Förderkennzeichen: WI 4086/2-1.

Literatur

[ABC+05] Michel Abdalla, Mihir Bellare, Dario Catalano, Eike Kiltz, Tadayoshi Kohno, Tanja Lange, John Malone-Lee, Gregory Neven, Pascal Paillier, and Haixia Shi. Searchable encryption revisited: Consistency properties, relation to anonymous ibe, and extensions. In *Advances in Cryptology–CRYPTO 2005*, pages 205–222. Springer, 2005.

[AKSX04] Rakesh Agrawal, Jerry Kiernan, Ramakrishnan Srikant, and Yirong Xu. Order preserving encryption for numeric data. In *Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, pages 563–574. ACM, 2004.

[API] Bouncy Castle Crypto API. www.bouncycastle.org.

[BCO11] Alexandra Boldyreva, Nathan Chenette, and Adam O’Neill. Order-preserving

encryption revisited: Improved security analysis and alternative solutions. In *Advances in Cryptology–CRYPTO 2011*, pages 578–595. Springer, 2011.

[BDCOP04] Dan Boneh, Giovanni Di Crescenzo, Rafail Ostrovsky, and Giuseppe Persiano. Public key encryption with keyword search. In *Advances in Cryptology-Eurocrypt 2004*, pages 506–522. Springer, 2004.

[DCdVFJ+11] Sabrina De Capitani di Vimercati, Sara Foresti, Sushil Jajodia, Stefano Paraboschi, and Pierangela Samarati. Private data indexes for selective access to outsourced data. In *Proceedings of the 10th annual ACM workshop on Privacy in the electronic society*, pages 69–80. ACM, 2011.

[G+03] Eu-Jin Goh et al. Secure indexes. *IACR Cryptology ePrint Archive*, 2003:216, 2003.

[HK14] Florian Hahn and Florian Kerschbaum. Searchable encryption with secure and efficient updates. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pages 310–320. ACM, 2014.

[Kal98] Burt Kaliski. Pkcs# 7: Cryptographic message syntax version 1.5. 1998.

[KPR12] Seny Kamara, Charalampos Papamanthou, and Tom Roeder. Dynamic searchable symmetric encryption. In *Proceedings of the 2012 ACM conference on Computer and communications security*, pages 965–976. ACM, 2012.

[KS14] Florian Kerschbaum and Axel Schroepfer. Optimal average-complexity ideal-security order-preserving encryption. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pages 275–286. ACM, 2014.

[Pai99] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Advances in cryptology, EUROCRYPT99*, pages 223–238. Springer, 1999.

[PRZB12] Raluca Ada Popa, Catherine Redfield, Nickolai Zeldovich, and Hari Balakrishnan. Cryptdb: Processing queries on an encrypted database. *Communications of the ACM*, 55(9):103–111, 2012.

[RVBM09] Mariana Raykova, Binh Vo, Steven M Bellovin, and Tal Malkin. Secure anonymous database search. In *Proceedings of the 2009 ACM workshop on Cloud computing security*, pages 115–126. ACM, 2009.

[Spi] SpiegelOnline. URL http://www.spiegel.de/netzwelt/gadgets/attacke-auf-playstation- netzwerk-hacker-stehlen-millionen-sony-kundendaten-a-759161.html (letzter Zugriff am 26.03.2015).

[SWP00] Dawn Xiaoding Song, David Wagner, and Adrian Perrig. Practical techniques for searches on encrypted data. In *Security and Privacy, 2000. S&P 2000. Proceedings. 2000 IEEE Symposium on*, pages 44–55. IEEE, 2000.

[Tan10] Qiang Tang. Privacy preserving mapping schemes supporting comparison. In *Proceedings of the 2010 ACM workshop on Cloud computing security workshop*, pages 53–58. ACM, 2010.

[Tan13] Qiang Tang. Search in encrypted data: Theoretical models and. *Theory and Practice of Cryptography Solutions for Secure Information Systems*, page 84, 2013.

[TKMZ13] Stephen Tu, M Frans Kaashoek, Samuel Madden, and Nickolai Zeldovich. Processing analytical queries over encrypted data. In *Proceedings of the VLDB Endowment*, volume 6, pages 289–300. VLDB Endowment, 2013.

[VDGHV10] Marten Van Dijk, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. Fully homomorphic encryption over the integers. In *Advances in cryptology–EUROCRYPT 2010*, pages 24–43. Springer, 2010.

[WWS05] Zheng-Fei Wang, Wei Wang, and Bai-Le Shi. Storage and query over encrypted character and numerical data in database. In *Computer and Information Technology, 2005. CIT 2005. The Fifth International Conference on*, pages 77–81. IEEE, 2005.

[XY12] Liangliang Xiao and I-Ling Yen. Security analysis for order preserving encryption schemes. In *Information Sciences and Systems (CISS), 2012 46th Annual Conference on*, pages 1–6. IEEE, 2012.