Karlsruher Institut für Technologie

Datenschutz und Privatheit in vernetzten Informationssystemen

Kapitel 11: Cloud Computing

Erik Buchmann (buchmann@kit.edu)

IPD, Systeme der Informationsverwaltung, Nachwuchsgruppe „Privacy Awareness in Information Systems“

Inhalte und Lernziele dieses Kapitels

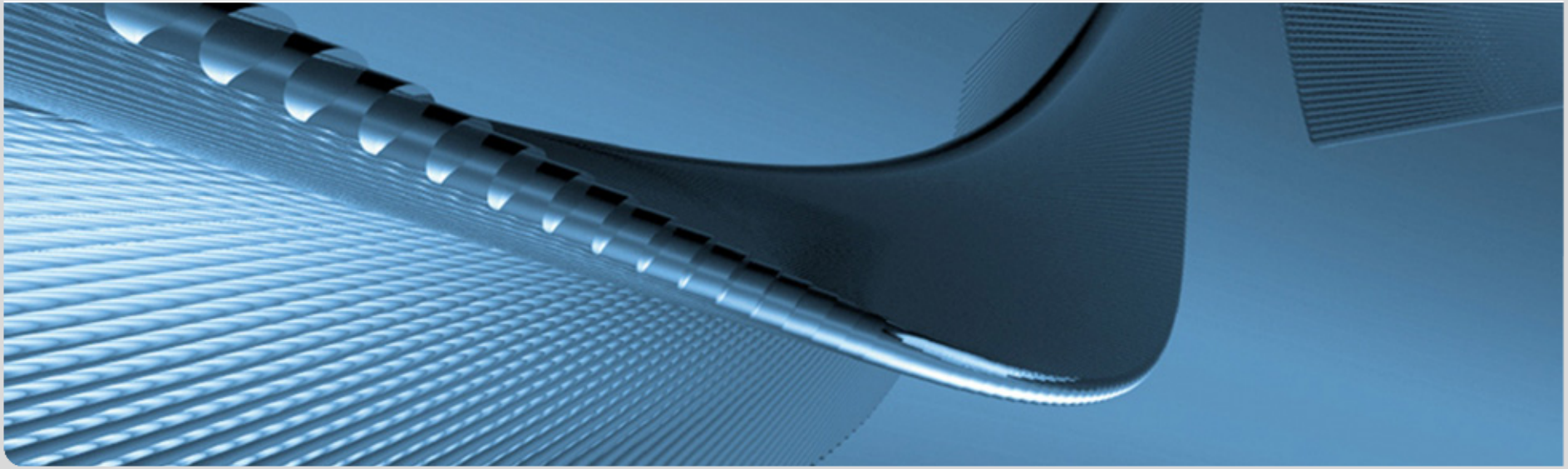
- Einführung: Cloud Computing
- Partially Order Preserving Encryption
- Homomorphe Verschlüsselung
- Abschluss

- Lernziele
  - Sie können erläutern, auf welche Weise Daten beim Cloud Computing gespeichert und verarbeitet werden.
  - Sie können die Datenschutzprobleme beim Cloud Computing erklären und von verwandten Problemstellungen wie Grid Computing oder Peer-to-Peer Computing abgrenzen.
  - Sie können einen Überblick darüber geben, welche aktuellen Techniken zum Datenschutz in Cloud-Computing Szenarien in der Forschung diskutiert werden, und welche Vor- und Nachteile diese haben.

Karlsruher Institut für Technologie

Cloud Computing

IPD, Systeme der Informationsverwaltung, Nachwuchsgruppe „Privacy Awareness in Information Systems“



Typische Cloud Computing-Probleme

- Ein Unternehmen möchte aus Gitternetzmodellen einen fotorealistischen 3D-Film berechnen.
- Ein Startup möchte CRM einführen, weiß aber nicht wieviele Kunden es in einem Jahr haben wird.
- Ein Nutzer möchte große Dateien im Web bereitstellen, kann aber nicht abschätzen, wieviele Personen parallel darauf zugreifen werden.

→ große Datenmengen
→ Anwendungen sind parallelisierbar
→ variable Hardwareanforderungen

Anm.: Dies sind auch die Unterschiede zum Verschlüsselte DB-Szenario, Kap. 10

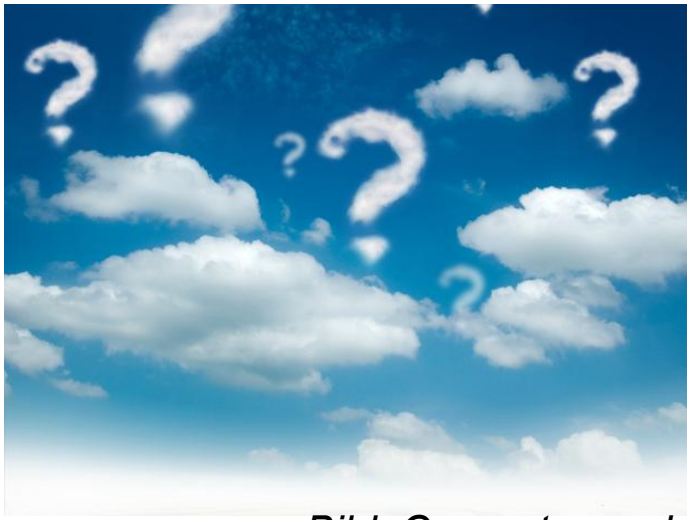


Bild: Computerwoche

Cloud Computing

- **Anwendungen, Daten, Infrastruktur** werden in ein verteiltes System ausgelagert
- **Endgeräte** haben nur noch Anzeigefunktion

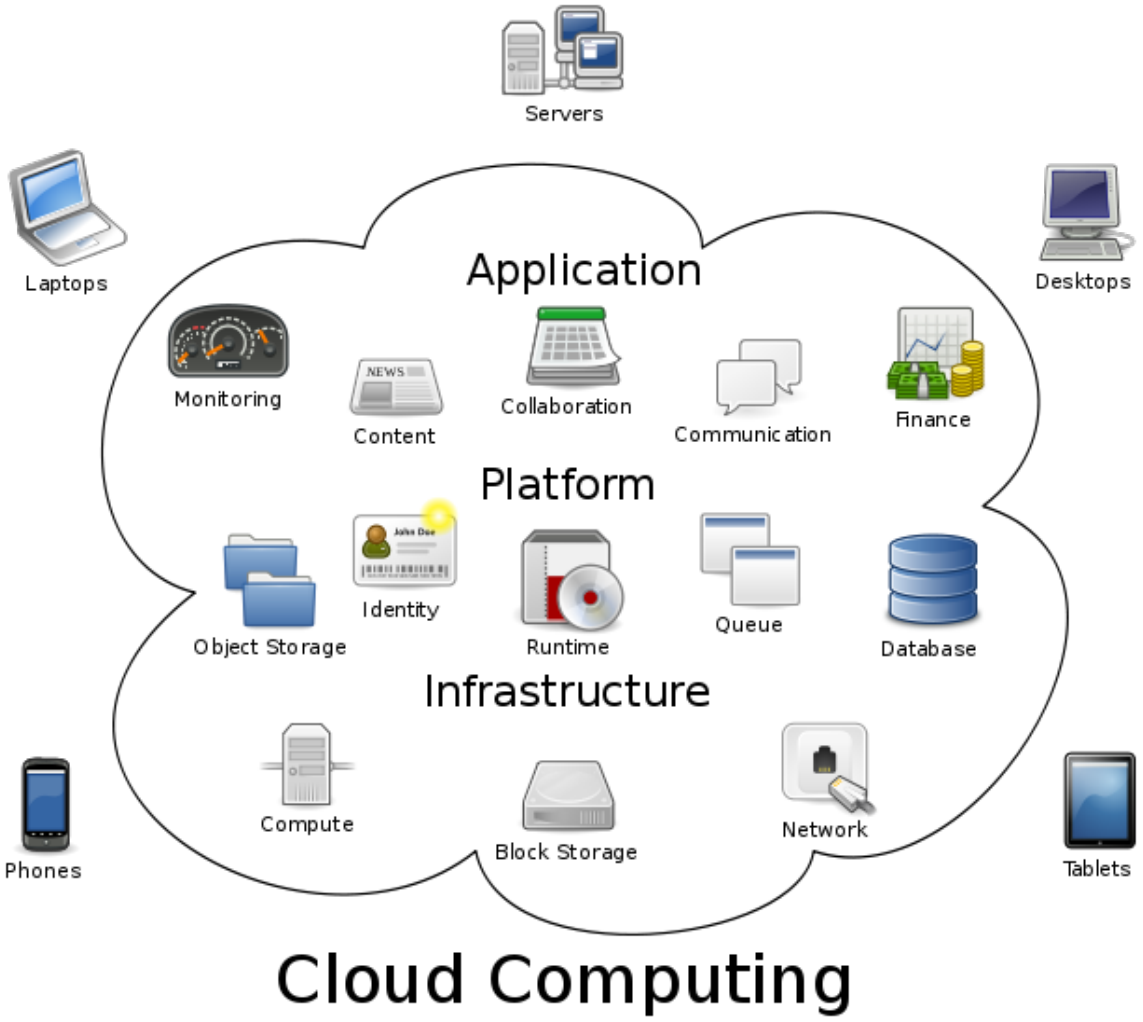


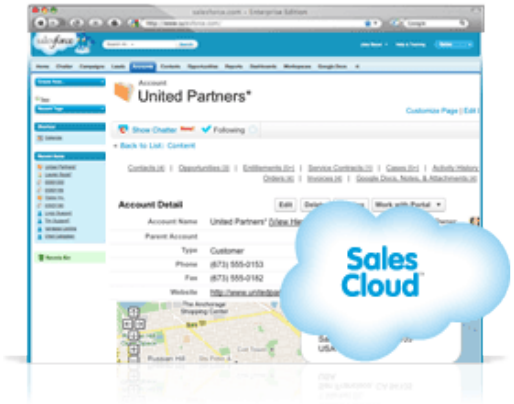
Bild: Wikimedia Commons

Ziele des Cloud Computing

- Verfügbarkeit
  - Ausfall einzelner Knoten egal; Anwendungen und Daten bleiben verfügbar
  - Zugriff über beliebige Endgeräte, von beliebigen Orten aus
- dynamische Skalierbarkeit
  - für Cloud Computing-Anwendungen ist es transparent, ob sie auf einem oder 100 Rechnern laufen
  - on demand: Cloud-Infrastruktur wächst mit der Last mit
- zentrales Anwendungsmanagement
  - Anwendungen vom Dienstleister installiert und gewartet
  - Dienstleister übernimmt Zugriffsschutz, Abrechnung, Updates etc.

Arten von Cloud-Dienstleistungen

- Software as a Service
  - Software installiert vom Service-Provider
  - Zugriff über Webbrowser vom Endgerät
  - z.B. CRM wie Salesforce



- Platform as a Service
  - Speicher, Backup, Zugriffsschutz bereitgestellt vom Provider
  - Zugriff über Anwendungen auf Endgerät
  - z.B. Amazon S3

amazon web services

- Infrastructure as a Service
  - Rechenleistung bereitgestellt vom Provider
  - Zugriff über Middleware-Frameworks wie Hadoop, MapReduce
  - z.B. Google App Engine

Beispiel für Middleware: Hadoop/MapReduce

- Hadoop (Apache) und MapReduce (Google) verteilen Rechenprozesse mit großen Datenmengen
  - Map: Aufteilung der Daten in (Schlüssel, Wert)-Paare, Berechnungen, lokaler Combine-Schritt
  - Reduce: zusammen-führen der Zwischen-ergebnisse gemäß der Schlüssel

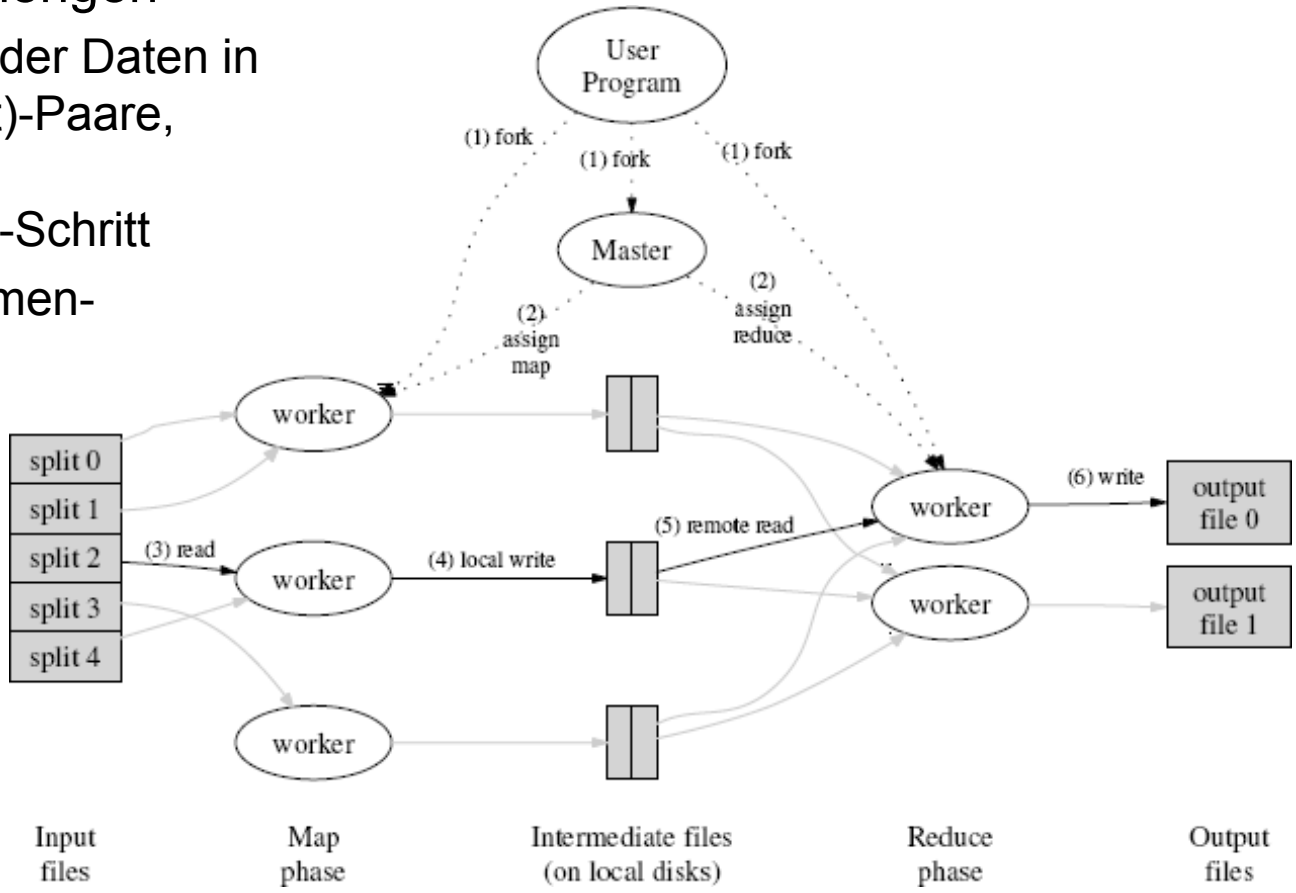


Bild: code.google.com

Abgrenzung zu anderen verteilten Systemen

- Peer-to-Peer Computing
  - Knoten von zahlreichen unabhängigen Organisationseinheiten betrieben
  - jedes Endgerät gleichzeitig Dienstleister und Dienstnehmer
- Grid Computing
  - virtueller Supercomputer aus unabhängigen Rechnernetzen
  - Middleware übernimmt Scheduling, Datenverteilung von Rechenaufgaben
  - losere Kopplung als bei klassischen Rechnerclustern
- Cloud Computing
  - zentraler Dienstleister (Microsoft, Amazon, Google, IBM, etc.) bietet *Dienstleistungen* „aus der Steckdose“
    - Speicher, Rechenleistung, Anwendungen
  - Endgeräte hauptsächlich als Betrachter

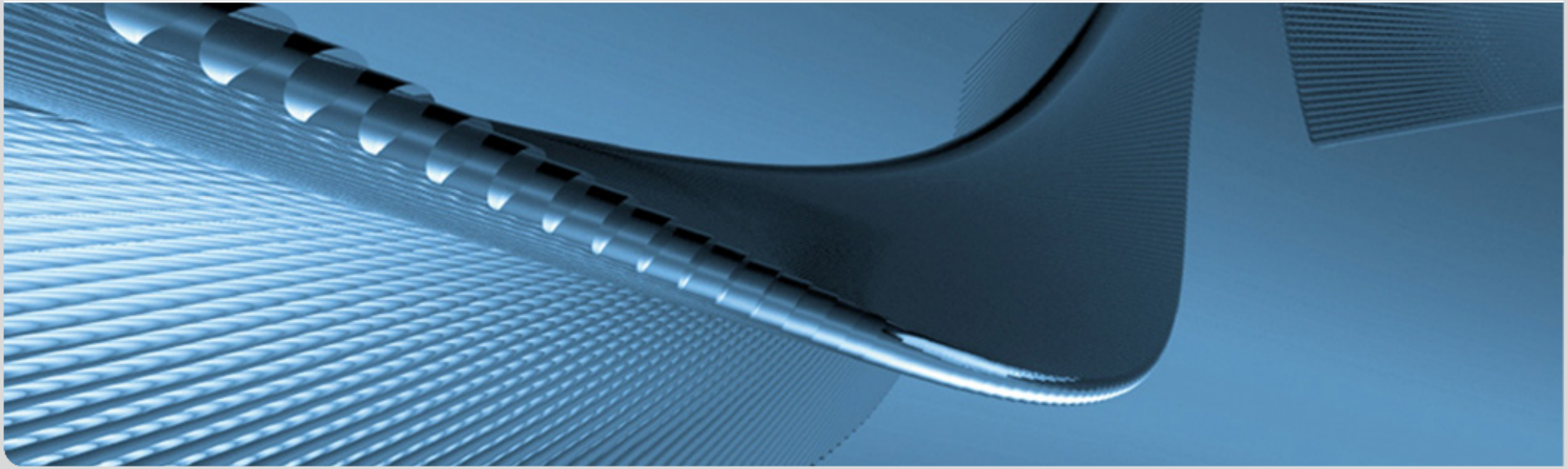
Datenschutz in der Cloud?

- Angreifermodell: Cloud-Anbieter „honest but curious“
  - arbeitet **korrekt** (kein Löschen, Manipulieren von Daten)
  - Ausspähen von **Daten** falls möglich
  - Ausspähen von **Anfragen** falls möglich
- Vergleichbare Probleme wie in allen verteilten Systemen
  - Daten liegen bei Fremdanbietern
  - Fremdanbieter sollen damit arbeiten, sie aber nicht einsehen können
    → Zugriffsschutz nicht ausreichend
- im Folgenden
  - Partially Order Preserving Encryption
  - Homomorphe Verschlüsselung

Karlsruher Institut für Technologie

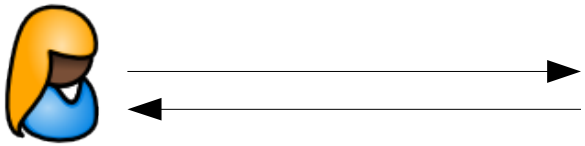
Partially Order Preserving Encryption

IPD, Systeme der Informationsverwaltung, Nachwuchsgruppe „Privacy Awareness in Information Systems“



Szenario

- Medizinische Datensätze für statistische Forschungen in der Cloud
- Anfrage:
  select count(*) from table
  where PLZ >= 76130 and PLZ < 76140
  and Krankheit = 'Magersucht'



PLZ	Krankheit
76127	Impotenz
76128	Hodenkrebs
76129	Sterilität
76131	Schizophrenie
76133	Diabetes
76133	Magersucht
76136	Magersucht

Wie die Anfrage in der Cloud berechnen, ohne dass der Anbieter die Daten sieht?

Funktioniert nicht: direkte Verschlüsselung

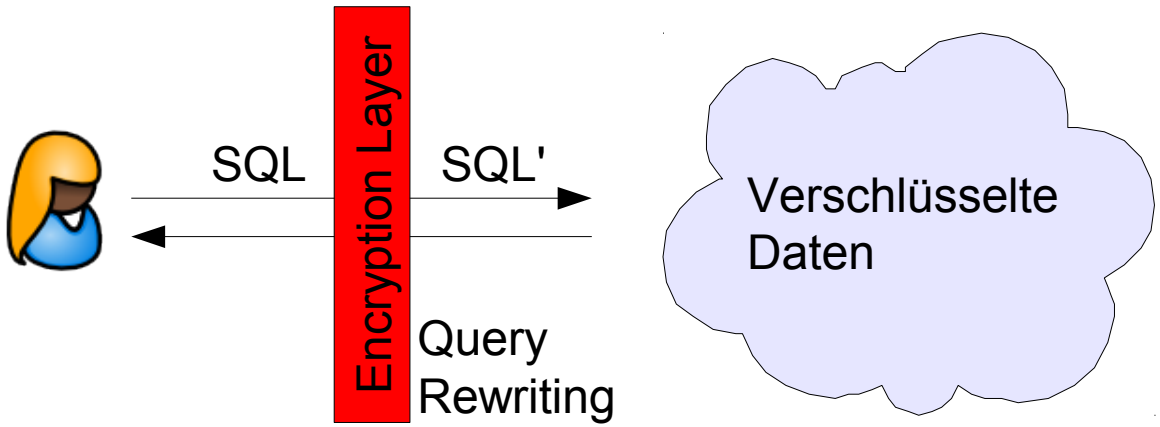
- Simpler Ansatz: Daten werden in der Cloud verschlüsselt gespeichert
- Kennt der Cloud-Anbieter den Schlüssel nicht, kann er die Anfrage nicht bearbeiten
- Kennt der Anbieter den Schlüssel, hat er auch Vollzugriff auf alle Daten

PLZ	Krankheit
5292	52967
2245	49732
7712	53025
9311	59232
1219	95623
3712	87324
6219	62423

- Anfrage:
  select count(*) from table
  where PLZ >= 76130 and PLZ < 76140
  and Krankheit = 'Magersucht'

Partially Order Preserving Encryption [1]

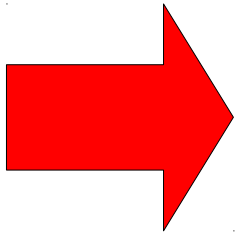
- Daten so auf kryptographische Codes abbilden, dass
  - Sortierreihenfolge der Codes gleich (ähnlich) der Reihenfolge der Originaldaten ist
  - Operationen vom Cloud-Anbieter **ohne Kenntnis des Schlüssels** auf den verschlüsselten Daten berechnet werden können
    - Selektionen (Exact Match, Bereichsanfragen mit <, ≤, etc.)
    - Gruppierungen, Verbund
    - Mengenoperationen (Union, Minus)
    - Aggregate (Min, Max, Count)



Grundlage: Order Preserving Encryption

- Abbildung der Daten auf ordnungserhaltende Codes

PLZ	Krankheit
76127	Impotenz
76128	Hodenkrebs
76129	Sterilität
76131	Schizophrenie
76133	Diabetes
76133	Magersucht
76136	Magersucht



K1S	AS9
15	100
16	80
29	154
56	153
70	53
70	104
71	104

- Umschreiben der Anfragen

```
select count(*) from table
where PLZ >= 76130
and PLZ < 76140
and Krankheit = 'Magersucht'
```

```
select count(*) from table
where K1S >= 30
and K1S < 100
and AS9 = 104
```

Order Preserving Encryption ist angreifbar

- Frequenzangriffe
  - *Hintergrundwissen*: Magersucht ist die häufigste Krankheit in Table
    → Magersucht hat Code 104
  - *Hintergrundwissen*: PLZ-Bezirk 76133 hat die meisten Einwohner
    → 76133 hat Code 70

- Domänenangriffe
  - *Hintergrundwissen*: Im Krankenhaus wird Sterilität und Schizophrenie behandelt; Magersucht hat Code 104
    → lexikalisch „S...“ > „M...“, also hat Schizophrenie Code 153 und Sterilität Code 154

K1S	AS9
15	100
16	80
29	154
56	153
70	53
70	104
71	104

Partially Order Preserving Encryption

- Unterteile das Data Set zufällig in mehrere Fragmente
  - Zahl der Fragmente bestimmt Dekodierungsaufwand
  - Zahl der Fragmente bestimmt Sicherheit gegen Angriffe
- Führe Order Preserving Encryption auf jedes Fragment aus
  - Ordnungen zwischen den Fragmenten unabhängig
- Beispiel für Spalte PLZ:

Original	1 Fragment	2 Fragmente	3 Fragmente	7 Fragmente
76127	1,15	1,15	3,1	1,0
76128	1,16	2,1	2,1	3,12
76129	1,29	1,29	1,2	7,12
76131	1,56	1,50	1,5	4,15
76133	1,70	2,70	2,7	2,12
76133	1,70	2,70	3,2	6,0
76136	1,71	1,70	1,7	5,7

Anfrageausführung

- Query Rewriting
  - Original
    select count(*) from table
    where PLZ >= 76130 and PLZ < 76140
  - Für 2 Fragmente
    select count(*) from table
    where (K1S >= 1,30 and K1S < 1,100)
    or (K1S >= 2,2 and K1S < 2,80)
- Aufwand der Anfrageausführung abhängig von der Zahl der Fragmente
  - Je mehr Fragmente, desto
    - mehr Metadaten muss der Verschlüsselungslayer speichern
    - aufwändiger die Anfrage
    - weniger funktionieren Indexe und Optimierungen der Datenbank

PLZ	K1S, 2 Fragmente
76127	1,15
76128	2,1
76129	1,29
76131	1,50
76133	2,70
76133	2,70
76136	1,70

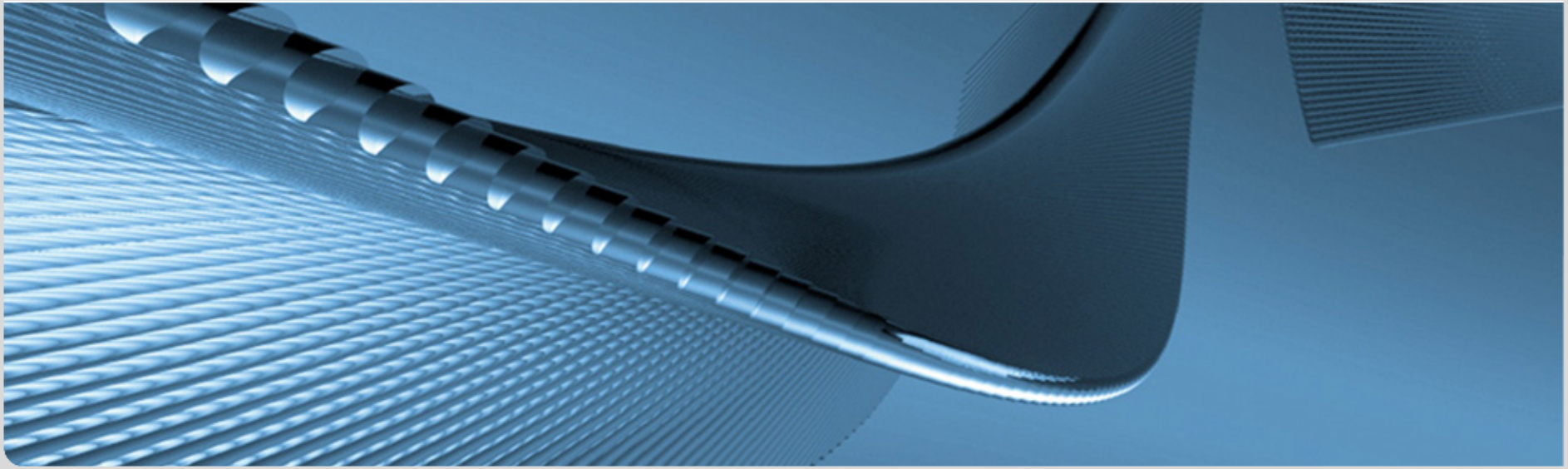
Diskussion

- Partially Order Preserving Encryption geeignet für viele Datenbankoperationen
  - Selektion, Verbund, Gruppierung, Mengenoperationen, Aggregate...
  - Nicht: arithmetische Operationen
- Einstellbarer Kompromiß zwischen Sicherheit und Performanz
  - nur ein Fragment: Order Preserving Encryption
  - soviele Fragmente wie Datensätze: herkömmliche Verschlüsselung
    → *Anwendungen, die Datenschutz UND Performanz brauchen?*
- Leicht realisierbar
  - keine Änderungen beim Cloud-Anbieter,
  - Encryption-Layer in der Middleware beim Dienstnutzer

Karlsruher Institut für Technologie

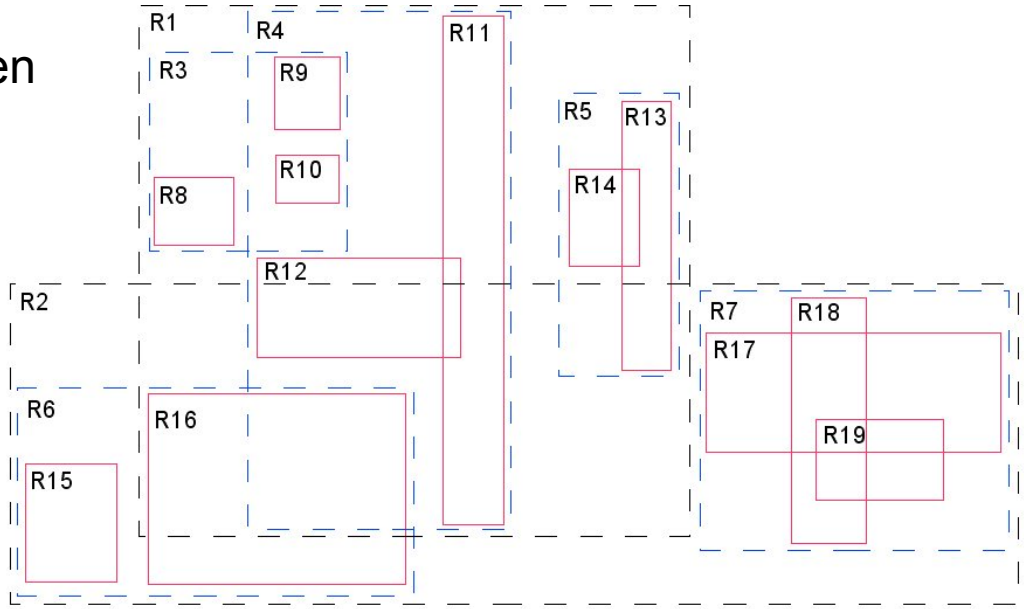
Homomorpe Verschlüsselung

IPD, Systeme der Informationsverwaltung, Nachwuchsgruppe „Privacy Awareness in Information Systems“



Szenario

- komplexe Rechenoperationen in der Cloud
  - Indexieren von räumlichen Daten (Vergleichs-operationen im R-Baum)
  - Clusteranalyse von Kundendaten (Abstandsmaße auf Vektoren)
  - etc.



$$dist^2(x, y) = \sum_{i=1}^{d} (x_i - y_i)^2$$

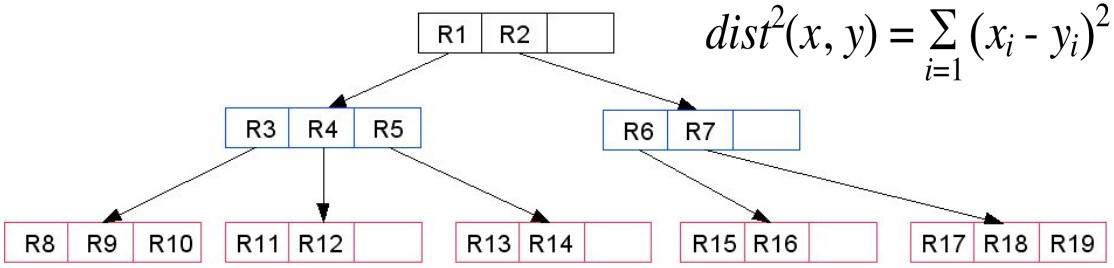


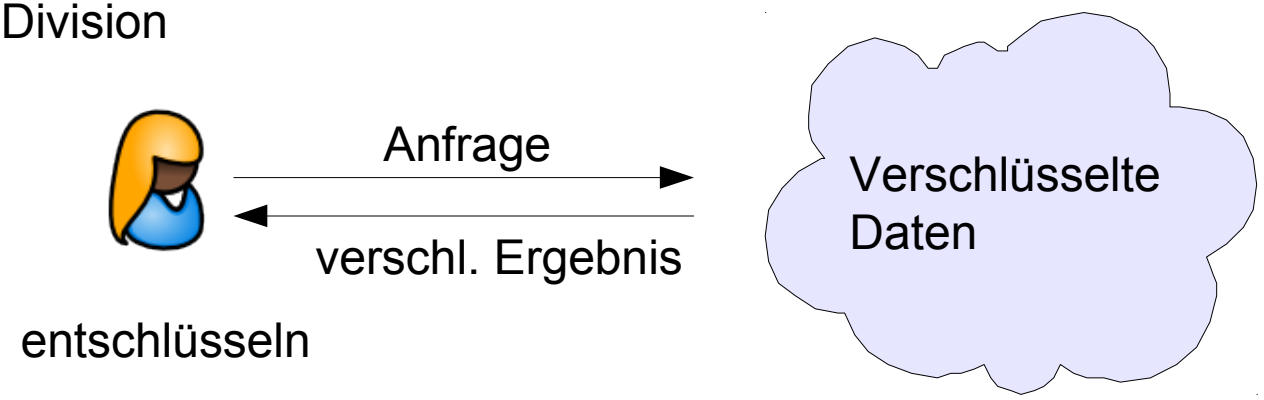
Bild: Wikimedia

Privacy Homomorphismus

- Daten so verschlüsseln, dass
  - Rechenoperationen **ohne Kenntnis des Schlüssels** auf verschlüsselten Daten ausgeführt werden
  - die Cloud verschlüsselte Ergebnisse an den Client übermittelt
  - der Client nach Entschlüsselung das Rechenergebnis erhält

- verschlüsselt mögliche Operationen (alle anderen Operationen muss Client berechnen)
  - Summe, Subtraktion
  - Multiplikation, Division

$$dist^2(x, y) = \sum_{i=1}^{d} (x_i - y_i)^2$$



Homomorphismus

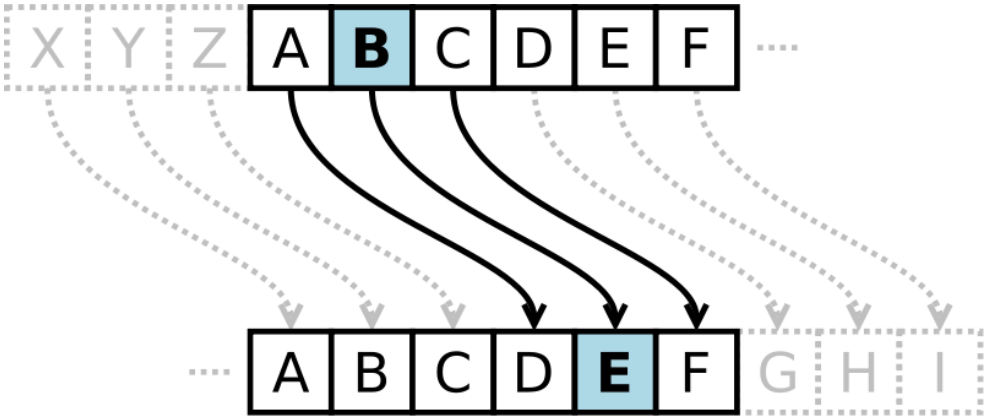
- Definition
  - zwei Verknüpfungen *p*, *q*,
    Abbildungsvorschrift *f*,
    Attribute $a_1 ... a_i$
  - *f* ist homomorph, wenn $f(p(a_1, a_2, \ldots a_i)) = q(f(a_1), f(a_2), \ldots f(a_i))$

- Beispiel: Addition
  - *f*, *q* seien Additionsoperationen,
    p ist eine Multiplikation mit 1,
    zwei Attribute a_1, a_2
  - *f* ist homomorph:
    $f(a_1 + a_2) = f(a_1) + f(a_2)$

→ Zentrale Frage: Welche Verschlüsselung ist für p, q geeignet und welche Abbildungsvorschrift unterstützt sie?

Beispiel: Caesar Chiffre

- Einfacher Ersetzungschiffre



- Caesar Chiffre ist **partiell homomorph**: unterstützt Konkatenation
  - Beispiel:

rot13(„Hello“) = „Uryyb“
rot13(„World“) = „Jbeyq“
Konkatenation(„Uryyb“, „Jbeyq“) = „UryybJbeyq“
rot13(„UryybJbeyq“) = „HelloWorld“

Bild: Wikimedia

Beispiel: RSA-Chiffre

- asymmetrische Public-Key-Verschlüsselung
  - Prinzip: Multiplikation von Primzahlen 'billiger' als Primzahlzerlegung

Klartext	k	öffentlicher Schlüssel	p*q, e
Primzahlen	p, q	privater Schlüssel	p*q, d
Ciphertext	$c := k^e$ MOD p*q	Entschlüsselung	$k := c^d$ MOD p*q

- RSA ist **partiell homomorph**: unterstützt Multiplikation
  Beispiel: verschlüsselte Berechnung von 7 * 2
  - Verschlüssele 7 und 2
    gewählt: p = 11, q = 13, berechnet: e = 23, d = 47
    enc(7) = (7^23) MOD (11*13) = 2
    enc(2) = (2^23) MOD (11*13) = 85
  - Multipliziere verschlüsselte Werte
    2 * 85 = 170
  - Entschlüssele Ergebnis
    dec(170) = (170^47) MOD (11*13) = 14

Hilfsvariablen:
$\varphi(p,q) = (p-1)*(q-1)$
e = zu $\varphi(p,q)$ teilerfremde Zahl
$d * e \equiv 1$ MOD $\varphi(p,q)$

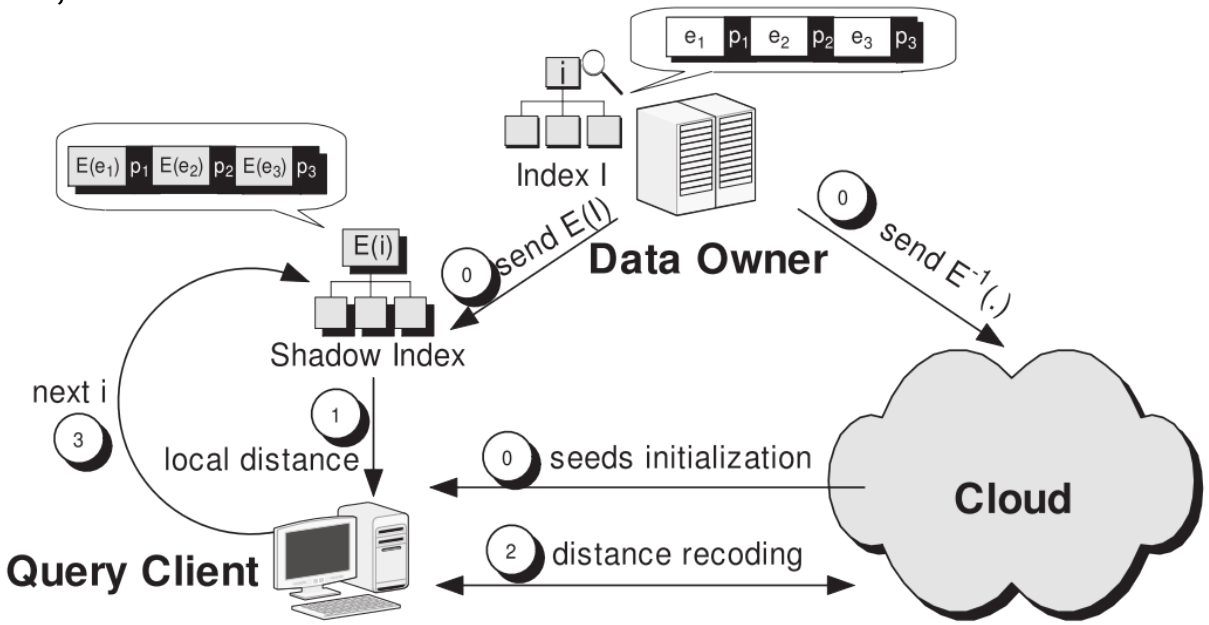
Volle Homomorphie nach [2]

- Verschlüsselung
  - Klartext k, geheime Primzahlen p, q
    Ciphertext (c,d) = enc(k) = [k MOD p, k MOD q]
- Operationen
  - Addition MOD p*q, Subtraktion MOD p*q, Multiplikation MOD p*q
  - alle anderen Operationen daraus herleitbar
  - auf dem Ciphertext (c,d) komponentenweise möglich
- Entschlüsselung
  - k = dec(c,d) durch den Chinesischen Restsatz
  - in polynomialer Zeit berechenbares Gleichungssystem

- *praktisches Problem*
  - Entschlüsselung sehr teure Operation,
    polynomial bezüglich Schlüssellänge

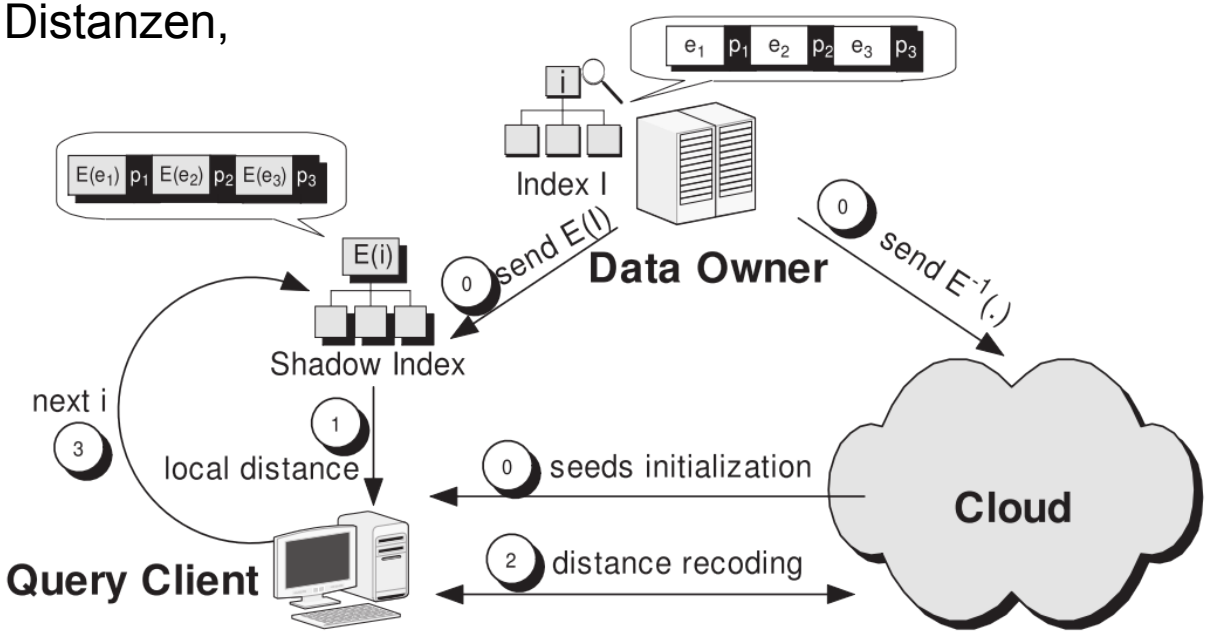
Beispiel: Private Distanz-Anfragen (1/3)

- Data Owner
  - speichert alle Schlüssel
  - speichert unverschlüsselten Index (R-Baum)
- Client
  - erhält vom Data Owner verschlüsselte Teile vom R-Baum (Shadow Index): Zeiger unverschlüsselt, Werte verschlüsselt
- Cloud
  - übernimmt Ver- und Entschlüsseln als teuerste Operation
  - *(könnte auch den Shadow Index senden)*



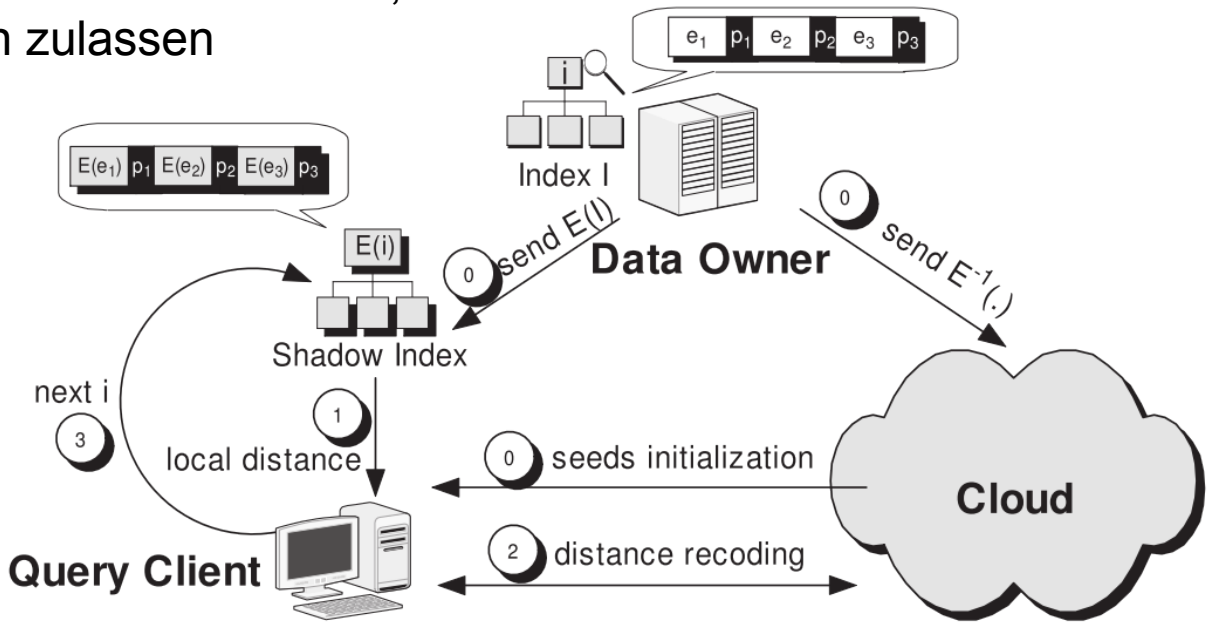
Beispiel: Private Distanz-Anfragen (2/3)

- Client erhält verschlüsselten Ausschnitt aus R-Baum (0)
- Traversierung des R-Baumes:
  - Client berechnet Distanzen zwischen eigenem Suchobjekt und den allen Index-Werten im aktuellen Knoten im Baum (1)
    - Berechnung auf verschlüsselten Werten
  - Client sendet verschlüsselte, permutierte Distanzen an Cloud
  - Cloud entschlüsselt Distanzen, rekodiert sie und sendet sie an Client (2)
  - Client wählt nächsten Knoten im Baum (3)
  - weiter mit (1)



Beispiel: Private Distanz-Anfragen (3/3)

- Data Owner
  - erfährt, wer Distanzanfragen im Baum durchführt
- Client
  - erfährt das Anfrageergebnis
  - Berechnung von Distanzen erfolgt verschlüsselt, Cloud teilt Client verfälschte Werte mit, die nur Auswahl von Knoten zulassen
- Cloud
  - erfährt Distanzen, aber nicht zwischen welchen Objekten diese berechnet werden

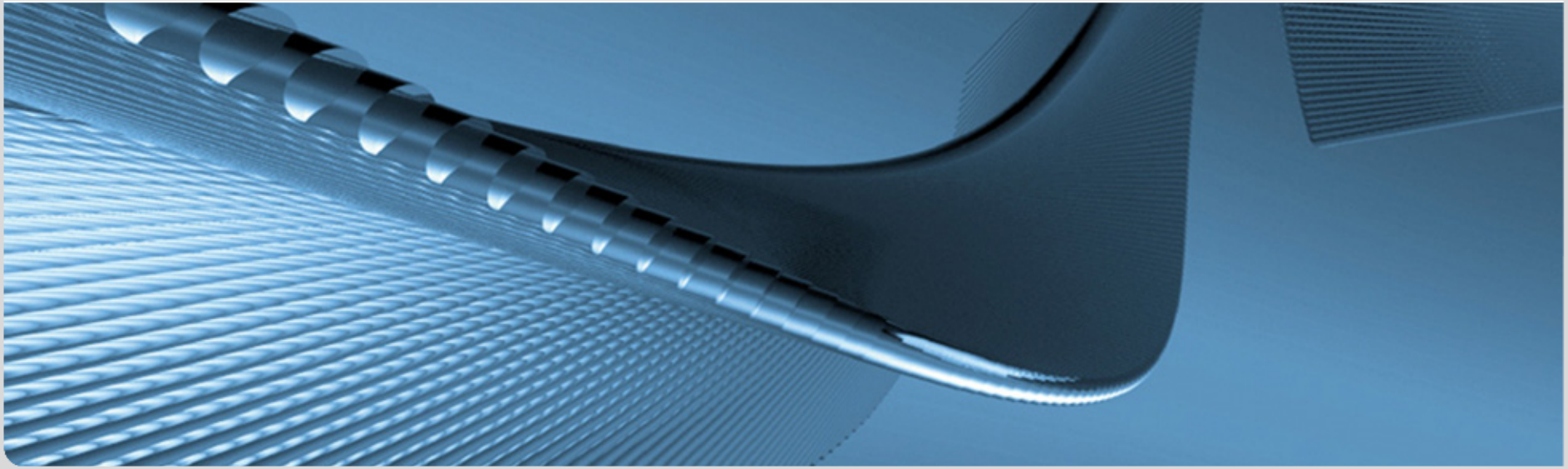


Diskussion

- Homomorphe Verschlüsselung ist berechnungsvollständig
  - im Prinzip alle arithmetischen Operationen durchführbar ohne Kenntnis der Originalwerte
  - komplexe Anwendungen sicher realisierbar
    - Beispiele: räumliche Indexierung, kNN-Suche, Clustering etc
- Aber: sehr teure Operation
  - einfache Additionen oder Multiplikationen bekommen polynomialen Aufwand
  - Ver- und Entschlüsselung in der Cloud (s. Beispiel) nur für wenige Szenarien geeignet

Karlsruher Institut für Technologie

Abschluss



Zusammenfassung

- Cloud Computing ist ein attraktives Anwendungsszenario
  - Software, Infrastruktur, Plattformdienste transparent virtualisiert, können mit Anforderungen mitwachsen
  - Aber: Datenschutzprobleme durch Auslagern von Personendaten
- Zwei technische Datenschutzansätze
  - Partially Order Preserving Encryption
    - für Datenbankoperationen geeignet
    - Kompromiß aus Datenschutz und Sicherheit
  - Homomorphe Verschlüsselung
    - für arithmetische Berechnungen geeignet
    - sehr berechnungsaufwendig

Literatur

[1] Stefan Hildenbrand et al.: *Query Processing on Encrypted Data in the Cloud,* Technical Report 735, ETH Zürich, 2011

[2] Haibo Hu, Jianliang Xu, Chushi Ren, Byron Choi: *Processing Private Queries Over Untrusted Data Cloud Through Privacy Homomorphism.* ICDE 2011