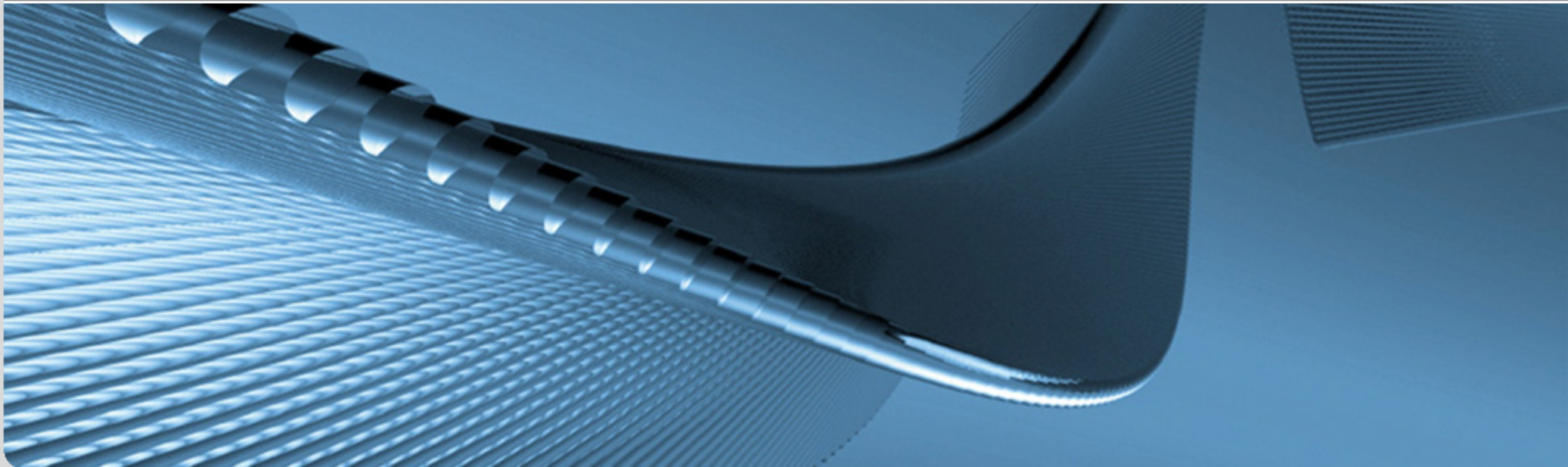


Datenschutz und Privatheit in vernetzten Informationssystemen

Kapitel 10: Verschlüsselte Datenbanken

Erik Buchmann (buchmann@kit.edu)

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



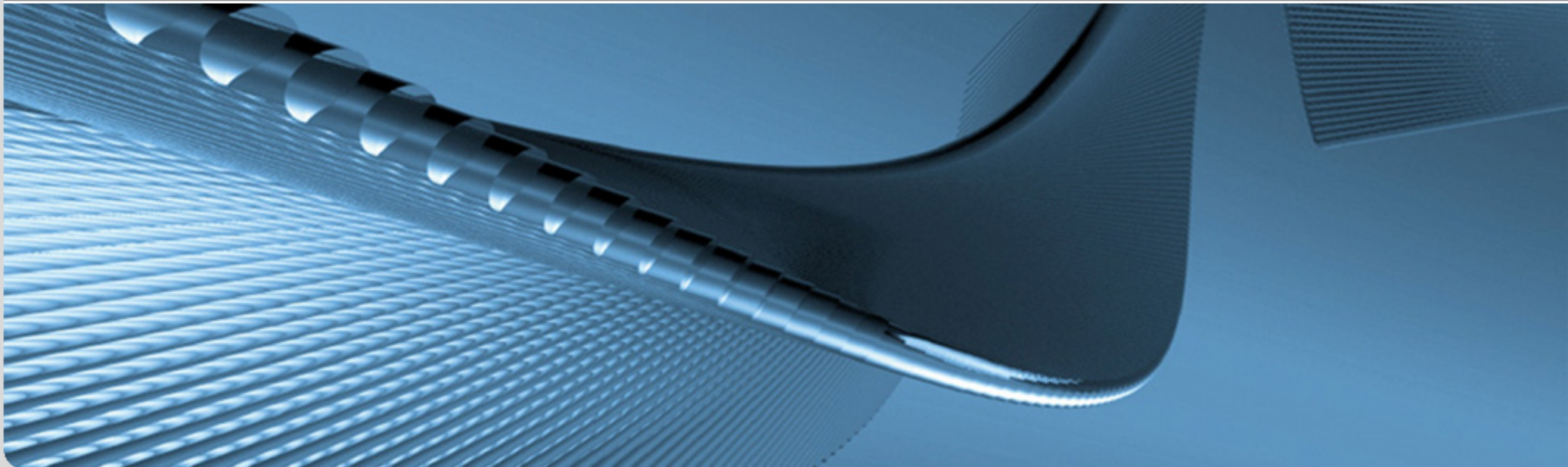
Inhalte und Lernziele dieses Kapitels

- Einführung: Encrypted Databases
- Fragmentation
- Bucketization
- Mehrdimensionale Bucketization
- Abschluss

- Lernziele
 - Sie können die Datenschutzprobleme erläutern, die sich durch das Auslagern von Datenbank-Funktionalität ergeben.
 - Sie können die vorgestellten Verfahren detailliert erläutern.
 - Anhand einer Vorgabe aus Datenschutzzielen, Anfragetypen und Performanzkriterien können Sie ein geeignetes Verfahren auswählen.

Encrypted Databases

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



Der Betrieb großer Datenbanken...

- ...ist nicht trivial
 - Regelmäßige Sicherheitsupdates und Backups
 - Diagnose, Wartung, Pflege, Tuning der Datenbank
 - Integration von neuen Anwendungen im laufenden Betrieb
 - Migration auf neue Hardware, Virtualisierungslösungen etc.
- ...ist nicht billig
 - Datenbank-Spezialisten zur Administration
 - Hardware mit genügend Reserveleistung für Lastspitzen
 - Personelle und technische Redundanzen gegen Systemausfall



*Rechenzentrum von
Facebook in Pineville*

Outsourcing von Datenbank-Diensten

- Dienstleister hat viele Kunden
 - Kann sich Spezialisten leisten
 - Reserve an Rechenleistung, Speicherkapazität, USV, Hardware etc. für mehrere Kunden ökonomischer
- Dienstleister kann in Haftung genommen werden
 - Verträge für garantierte Leistungen
- *Aber: Vertraulichkeit der Daten?*
 - Unternehmensdaten auf fremden Servern,
 - ggf. im Ausland
→ Datenschutzpflichten



Architekturmodell

- Dienstleister verwaltet Datenbank für den Kunden
 - Speichert Daten
 - Führt Anfragen (SQL) vom **Client** aus
 - **Server** liefert Anfrageergebnisse zurück

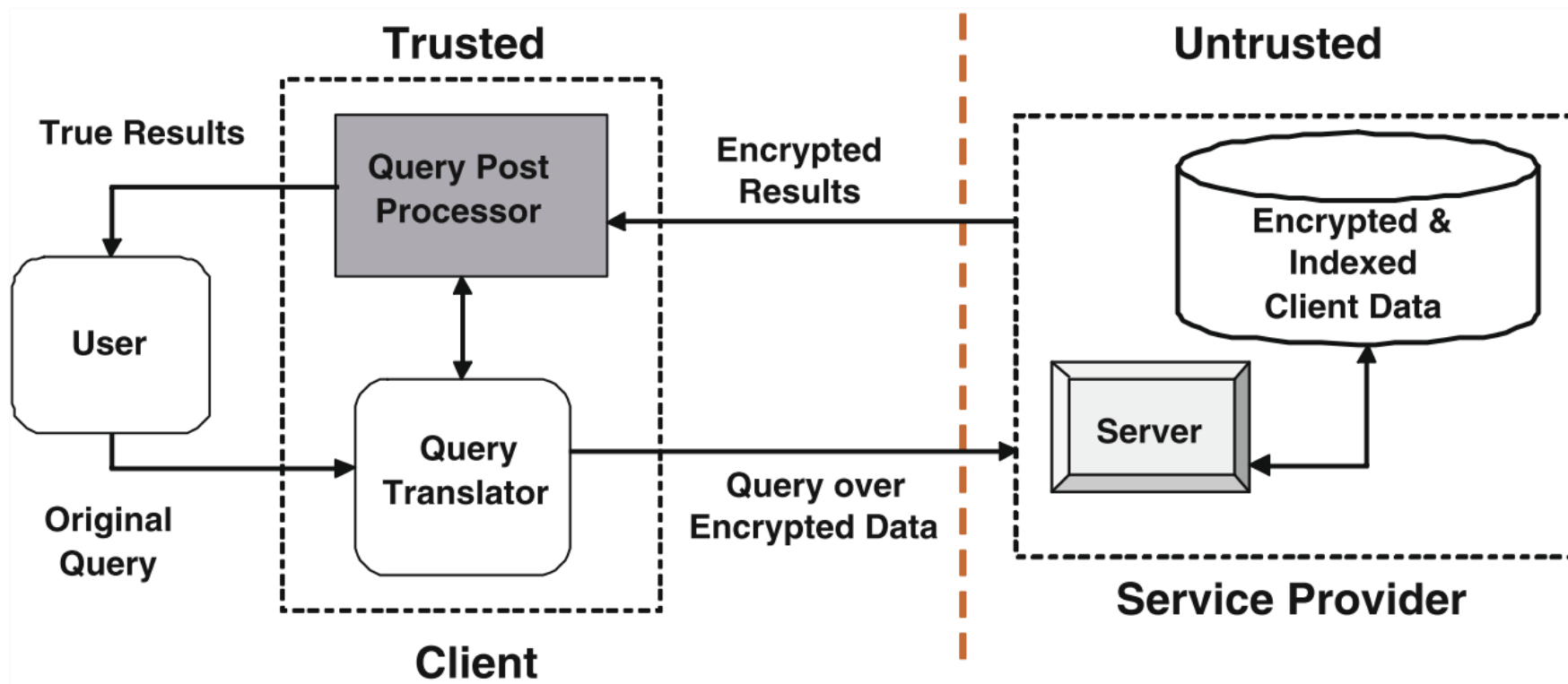
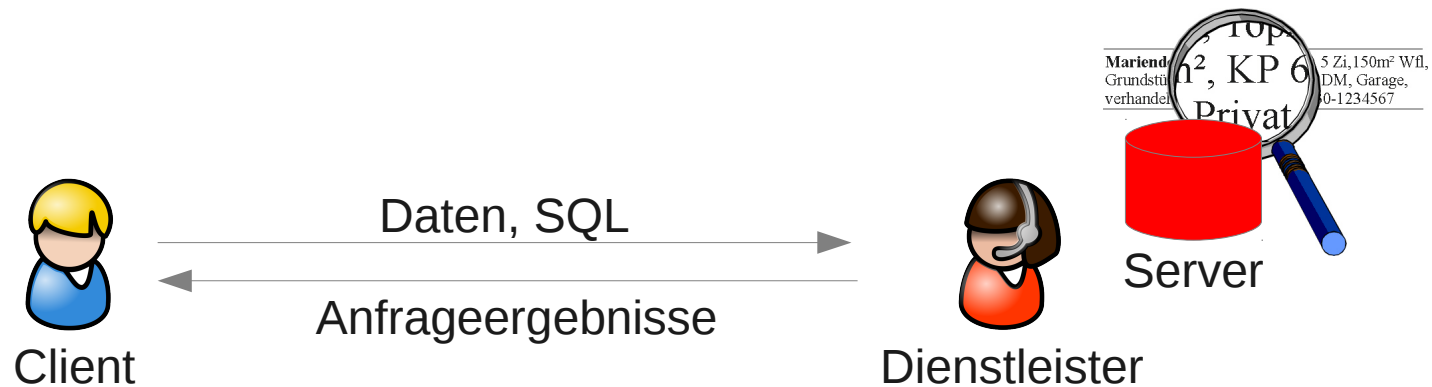


Bild: [3]

Angreifermodell

- Dienstleister ist **Honest-but-Curious**
 - Dienste ordnungsgemäß leisten, aber
 - sensitive Informationen sammeln
 - Datenbank, Anfragen und Ergebnisse ausspähen
 - Zugriffszeiten und -muster mitprotokollieren
 - **Frequenz- und Zuordnungsangriffe** durchführen



Angriffe (Wiederholung)

Zuordnungsangriffe

- Ausprobieren, ob man den Klartext erraten kann
- Beispiel: Befindet sich „Alice“ in einer verschlüsselten DB?
 - Angreifer kennt Public Key: „Alice“ → „H71A00EF91“

Person	Krankheit	Alter
AA73F9E001	Magersucht	24
H71A00EF91	Diabetes	31
3429A9F64E	Schnupfen	26

Frequenzangriffe

- Aus der Verteilung der Ciphertexte kann auf Inhalte geschlossen werden
- Beispiel: Hat „Alice“ in verschlüsselter Datenbank Urlaub genommen?
 - Angreifer weiß, dass am Jahresanfang „30 Tage“ häufigster Eintrag

Person	Urlaubstage
Alice	A7
Bob	EF
Carol	A7
Dave	A7
Eve	21
Frank	FF

Anfragetypen

Exact-Match-Anfragen

```
SELECT * FROM db WHERE Diagnose='Schnupfen'
```

(Mehrdimensionale) Bereichsanfragen

```
SELECT * FROM db WHERE PLZ>76221 AND PLZ<76225
```

Verbundanfragen

```
SELECT * FROM db AS a, db AS b WHERE a.Name=b.Name AND
a.Diagnose='Schnupfen' and b.Diagnose='Heiser'
```

Mengenoperationen

```
SELECT * FROM db WHERE Diagnose='Schnupfen'
MINUS SELECT * FROM db WHERE Diagnose='Heiser'
```

Aggregate (Min, Max, Avg, Count)

```
SELECT AVG(Gbt) FROM db
WHERE Diagnose='Schnupfen'
```

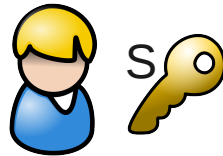
Gruppierung

```
SELECT AVG(Gbt) FROM db
GROUP BY Diagnose
HAVING AVG(Gbt) > 1985
```

db	Name	Gbt	PLZ	Diagnose
	Alice	1981	76227	Schnupfen
	Bob	1974	76221	Husten
	Carol	1995	76221	Heiser
	Dave	2001	76229	Schnupfen
	Alice	1981	76227	Husten

Lösung: Daten verschlüsseln?

- Client verschlüsselt die Daten, die der Dienstleister auf Server speichert



Name	PLZ	Diagnose
Alice	76227	Schnupfen
Bob	76221	Husten
Carol	76221	Heiser
Dave	76229	Schnupfen
Alice	76227	Husten



Anzr	CYM	Qvntabfr
Nyvpr	21772	Fpuahcsra
Obo	21776	Uhfgra
Pneby	21776	Urvfre
Qnir	21774	Fpuahcsra
Nyvpr	21772	Uhfgra

- Welche Arten von Anfragen effizient auf Server ausführbar?
- Ist das Schema angreifbar?

Performance vs. begrenzte Offenlegung

- Guter Datenschutz
 - Starke Verschlüsselung ist langsam
 - Insbes. probabilistische Verschlüsselung
 - Hoher Aufwand gegen Frequenz- und Zuordnungsangriffe
 - Dienstleister kann Anfragen und Zugriffe auf Server mitloggen
 - Verschlüsselungsschema kann manche Anfragen sehr teuer machen
 - Wenn große Teile der DB vom Server heruntergeladen und vom Client entschlüsselt werden müssen

- Gute Performanz
 - Erzwingt einfache Verfahren, Risiko für Offenlegung steigt
 - Anfrageausführung auf dem Server
 - Wenn Server den Schlüssel benötigt, ist nichts gewonnen

Anforderungen

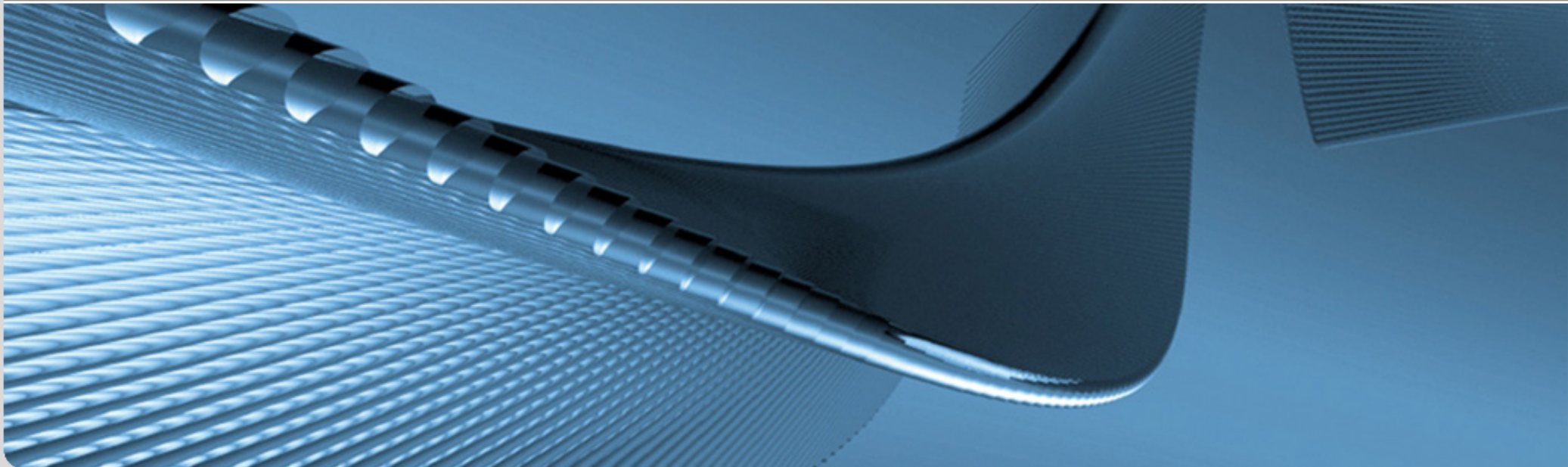
- R1: Begrenzte Offenlegung
 - Es muss klar werden, wieviel der Dienstleister maximal aus dem Datenbestand lernen kann

- R2: Zugriffsregeln
 - Datenbesitzer muss steuern können, wer welche Daten einsehen darf
 - Gilt ggf. auch für Anfrageergebnisse
 - Dienstleister soll nicht erfahren, was er berechnet

- R3: Performanz
 - Ausführungsgeschwindigkeit muss „akzeptabel“ bleiben, also nicht $O(n^k)$ oder dergl.

Fragmentation

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



Schutzziel

- Angreifermodell
 - Angreifer ist **Honest-but-Curious**
 - Angreifer hat Zugang zur Datenbank (Outsourcing-Szenario)

- Angreifer will
 - Zuordnung von sensitiven Attributen erfahren, z.B. VersNr-Name, Name-Diagnose etc.
 - Attributverteilung ist egal, z.B. die in der DB enthaltenen Diagnosen → kein Schutz gegen Frequenzangriffe

VersNr	Name	Gbt	PLZ	Arzt	Diagnose
123	Alice	1981	76227	Dr. Brown	Schnupfen
456	Bob	1974	76221	Dr. Red	Husten
789	Carol	1995	76221	Dr. Olive	Heiser
101	Dave	2001	76229	Dr. Blue	Schnupfen
112	Alice	1981	76227	Dr. Red	Husten

Systemmodell

■ Datenbesitzer legt Vertraulichkeitsregeln (Constraints) fest

■ Welche Attribute und Zuordnungen nicht offenlegen?



VersNr	Name	Gbt	PLZ	Arzt	Diagnose
123	Alice	1981	76227	Dr. Brown	Schnupfen
456	Bob	1974	76221	Dr. Red	Husten
789	Carol	1995	76221	Dr. Olive	Heiser

- $C_0: \{\text{VersNr}\}$
- $C_1: \{\text{Name, Gbt}\}$
- $C_2: \{\text{Name, PLZ}\}$
- $C_3: \{\text{Name, Diagnose}\}$
- $C_4: \{\text{Name, Arzt}\}$
- $C_5: \{\text{Gbt, PLZ, Diagnose}\}$
- $C_6: \{\text{Gbt, PLZ, Arzt}\}$

■ Dienstleister erhält

■ Teilweise verschlüsselte Fragmente der Datenbank, die nur vom Schlüsselbesitzer zusammenzuführen sind [2]



F_1			F_2				F_3			
<u>salt</u>	enc	Name	<u>salt</u>	enc	Gbt	PLZ	<u>salt</u>	enc	Arzt	Diagnose
S_1	#####	Alice	S_4	#####	1981	76227	S_7	#####	Dr. Brown	Schnupfen
S_2	#####	Bob	S_5	#####	1974	76221	S_8	#####	Dr. Red	Husten
S_3	#####	Carol	S_6	#####	1995	76221	S_9	#####	Dr. Olive	Heiser

Constraints

- Relationenschema $R = \{A_0, A_1, \dots, A_n\}$,

Constraint $c \subseteq R$

- $|c| = 1$ (nur 1 Attribut): **Attribut** nicht veröffentlichen (c_0)
- $|c| > 1$: **Assoziation** zwischen den Attributen nicht veröffentlichen ($c_1 \dots c_6$)
- alle anderen Attribute und Attributkombinationen veröffentlichsbar (z.B. {Arzt, Gbt} oder {Diagnose, PLZ} sind erlaubt)

- **Wohldefiniertheit** einer Menge von Constraints $C = \{c_0, c_1, \dots, c_k\}$

- Constraints dürfen sich nicht überschneiden

$$\forall c_i, c_j \in C: i \neq j \wedge c_i \not\subseteq c_j$$

db

VersNr	Name	Gbt	PLZ	Arzt	Diagnose
123	Alice	1981	76227	Dr. Brown	Schnupfen
456	Bob	1974	76221	Dr. Red	Husten
789	Carol	1995	76221	Dr. Olive	Heiser

C_0 : {VersNr}

C_1 : {Name,Gbt}

C_2 : {Name,PLZ}

C_3 : {Name,Diagnose}

C_4 : {Name,Arzt}

C_5 : {Gbt,PLZ,Diagnose}

C_6 : {Gbt,PLZ,Arzt}

Fragmente

- Relationenschema $R = \{A_0, A_1, \dots, A_n\}$, Constraints $C = \{c_0, c_1, \dots, c_k\}$
- Jedes Fragment speichert
 - **salt**
 - beliebige Zufallszahl im Klartext, Primärschlüssel in R
 - **Sichtbare Attribute**
 - Nicht-sensible Attribute
 $\{A_i, \dots, A_j\} \subseteq R$
 - **enc**
 - Verschlüsselung aller restlichen Attribute eines Tupels XOR salt
 $enc = encode([R - \{A_i, \dots, A_j\}] \otimes salt)$

F_1	<u>salt</u>	enc	Name
	124	encode({123, 1981, 76227, Dr. Brown, Schnupfen} \otimes 124)	Alice
	893	encode({456, 1974, 76221, Dr. Red, Husten} \otimes 893)	Bob
	410	encode({789, 1995, 76221, Dr. Olive, Heiser} \otimes 410)	Carol

Anfrageverarbeitung auf Fragmenten

- Query Rewriting auf passendes Fragment
 - SELECT Diagnose FROM db WHERE Name = 'Alice' AND PLZ = 76227
 - SELECT salt, enc FROM F_1 WHERE Name = 'Alice'
 - oder SELECT salt, enc FROM F_2 WHERE PLZ = 76227
 - oder SELECT salt, enc FROM F_3 (je nach Selektivität des Fragments)
- Client dekodiert Ergebnis vom Server
 - dec = decode(enc) \otimes salt
- Client berechnet Anfrageergebnis
 - SELECT Diagnose FROM dec WHERE Name = 'Alice' AND PLZ = 76227

F_1	<u>salt</u>	enc	Name
	124	encode({123, 1981, 76227, Dr. Brown, Schnupfen} \otimes 124)	Alice
	893	encode({456, 1974, 76221, Dr. Red, Husten} \otimes 893)	Bob
	410	encode({789, 1995, 76221, Dr. Olive, Heiser} \otimes 410)	Carol

Optimale Fragmentierung

- Korrektheit
 - Keine Informationspreisgabe bezüglich Constraints

- Maximale Sichtbarkeit
 - Möglichst viele Attribute im Klartext
 - Ausfiltern von unnötigen Zwischenergebnissen ohne Entschlüsselung

- Minimale Fragmentierung
 - Möglichst wenig Fragmente mit nur wenigen Attributen
 - Anfragen mit mehreren Attributen effizient ausführen

Korrektheit

- geg.: Relation $R = \{A_0, A_1, \dots, A_n\}$,
Constraints $C = \{c_0, c_1, \dots, c_k\}$

- Korrektheit der Fragmentierung

$$F = \{F_0, F_1, \dots, F_k\}$$

- Kein Fragment mit verbotenen Attributkombinationen

$$\forall f \in F, \forall c \in C: c \not\subseteq f$$

- Fragmente haben keine gemeinsamen Attribute (Verknüpfbarkeit)

$$\forall p, q \in F: p \cap q = \emptyset$$

- Zahlreiche korrekte Fragmentierungen, welche ist die beste?

- $\{\}$
- $\{\text{Name}\}$
- $\{\text{Gbt}, \text{PLZ}\}$
- $\{\text{PLZ}, \text{Arzt}, \text{Diagnose}\}$
- $\{\text{Name}\}, \{\text{Gbt}, \text{Arzt}, \text{Diagnose}\}, \{\text{PLZ}\}$
- $\{\text{Name}\}, \{\text{Gbt}, \text{PLZ}\}, \{\text{Arzt}, \text{Diagnose}\}$
- $\{\text{Name}\}, \{\text{Gbt}\}, \{\text{PLZ}, \text{Arzt}, \text{Diagnose}\}$
- $\{\text{Name}\}, \{\text{Gbt}\}, \{\text{PLZ}\}, \{\text{Arzt}\}, \{\text{Diagnose}\}$

R: {	VersNr,	C_0 : {VersNr}
	Name,	C_1 : {Name,Gbt}
	Gbt,	C_2 : {Name,PLZ}
	PLZ,	C_3 : {Name,Diagnose}
	Arzt,	C_4 : {Name,Arzt}
	Diagnose}	C_5 : {Gbt,PLZ,Diagnose}
		C_6 : {Gbt,PLZ,Arzt}

Maximale Sichtbarkeit

■ geg.: Relation $R = \{A_0, A_1, \dots, A_n\}$,
Fragmente $F = \{F_0, F_1, \dots, F_k\}$

■ viele Attribute im Klartext

■ Client muss kleinere Zwischenergebnisse entschlüsseln,
wenn Server Tupel ausfiltern kann

■ Alle Attribute in R , die nicht in einem einelementigen Constraint
enthalten sind, sind Bestandteil eines Fragments

■ $\forall a \in R, \{a\} \notin C: \exists f \in F$ so dass $a \in f$

■ Welche Fragmentierung ist optimal?

~~■ $\{\}$~~

~~■ $\{\text{Name}\}$~~

~~■ $\{\text{Gbt}, \text{PLZ}\}$~~

~~■ $\{\text{PLZ}, \text{Arzt}, \text{Diagnose}\}$~~

■ $\{\text{Name}\}, \{\text{Gbt}, \text{Arzt}, \text{Diagnose}\}, \{\text{PLZ}\}$

■ $\{\text{Name}\}, \{\text{Gbt}, \text{PLZ}\}, \{\text{Arzt}, \text{Diagnose}\}$

■ $\{\text{Name}\}, \{\text{Gbt}\}, \{\text{PLZ}, \text{Arzt}, \text{Diagnose}\}$

■ $\{\text{Name}\}, \{\text{Gbt}\}, \{\text{PLZ}\}, \{\text{Arzt}\}, \{\text{Diagnose}\}$

R: { VersNr,
Name,
Gbt,
PLZ,
Arzt,
Diagnose}

C_0 : {VersNr}

C_1 : {Name,Gbt}

C_2 : {Name,PLZ}

C_3 : {Name,Diagnose}

C_4 : {Name,Arzt}

C_5 : {Gbt,PLZ,Diagnose}

C_6 : {Gbt,PLZ,Arzt}

Minimale Fragmentierung

- Je weniger Fragmente, desto besser
 - Server kann eventuell Anfragen mit mehr als einem Attributen bearbeiten
`SELECT * FROM db WHERE Arzt = 'Dr. Red' AND Diagnose='Husten'`

- geg.: zwei korrekte Fragmentierungen F, F' mit max. Sichtbarkeit

- F ist minimal wenn kein **korrektes** F' mit **max. Sichtbarkeit** existiert, das weniger Fragmente enthält
 - es ist unmöglich, aus zwei Fragmenten in F eines in F' zu bauen
 - $\nexists F'$ so dass $|F'| < |F|$

- *Es kann mehrere minimale Fragmentierungen geben*

- {}
- {Name}
- {Gbt, PLZ}
- {PLZ, Arzt, Diagnose}

- {Name}, {Gbt, Arzt, Diagnose}, {PLZ}
- {Name}, {Gbt, PLZ}, {Arzt, Diagnose}
- {Name}, {Gbt}, {PLZ, Arzt, Diagnose}
- {Name}, {Gbt}, {PLZ}, {Arzt}, {Diagnose}

Query Performance

- Anfragen an einzelne sichtbare Attribute
 - Performanz besser als Bucketization
 - Kein Entschlüsseln von „false Positives“, d.h. nicht gesuchten Attributen im gleichen Bucket
 - Bereichsanfragen, Gruppierung, Sortierung etc. möglich

- Selektionsanfragen auf unsichtbare Attribute, Selektionen über mehrere Attribute, Verbundanfragen
 - Client muss ganze Fragmente herunterladen und lokal entschlüsseln

F ₁			F ₂				F ₃			
<u>salt</u>	enc	Name	<u>salt</u>	enc	Gbt	PLZ	<u>salt</u>	enc	Arzt	Diagnose
S ₁	#####	Alice	S ₄	#####	1981	76227	S ₇	#####	Dr. Brown	Schnupfen
S ₂	#####	Bob	S ₅	#####	1974	76221	S ₈	#####	Dr. Red	Husten
S ₃	#####	Carol	S ₆	#####	1995	76221	S ₉	#####	Dr. Olive	Heiser

Diskussion Fragmentation

■ Vorteile

- Detaillierte Schutzziele definierbar
- Einfach einzusetzen, keine großen Änderungen an existierenden DBMS
- Sehr gute Performance bei Anfragen auf einzelne sichtbare Attribute

■ Nachteile

- Relativ niedrige Schutzziele, Attributverteilung etc. offen
- Verbundanfragen oder Anfragen über mehrere Fragmente erfordern herunterladen und entschlüsseln des kompletten Fragments

F ₁			F ₂				F ₃			
<u>salt</u>	enc	Name	<u>salt</u>	enc	Gbt	PLZ	<u>salt</u>	enc	Arzt	Diagnose
S ₁	#####	Alice	S ₄	#####	1981	76227	S ₇	#####	Dr. Brown	Schnupfen
S ₂	#####	Bob	S ₅	#####	1974	76221	S ₈	#####	Dr. Red	Husten
S ₃	#####	Carol	S ₆	#####	1995	76221	S ₉	#####	Dr. Olive	Heiser

Bucketization

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



Schutzziel

- Angreifermodell
 - Angreifer ist **Honest-but-Curious**
 - Angreifer hat Zugang zur Datenbank (Outsourcing-Szenario)
- Angreifer will
 - Vollständigen Inhalt der Datenbank offenlegen
 - Zuordnung von Individuen zu sensitiven Attributen erfahren (Frequenz- und Zuordnungsangriffe)

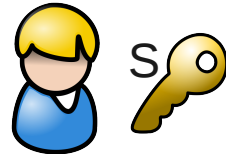
Name	PLZ	Diagnose
Alice	76227	Schnupfen
Bob	76221	Husten
Carol	76221	Heiser
Dave	76229	Schnupfen
Alice	76227	Husten

Systemmodell

- Server speichert

- Datenbank db mit Relationenschema R,
tupelweise verschlüsselt: $enc = encode(\{A_0, \dots, A_n\}), A_0, \dots, A_n \in R$

- kodierten Index



db

Name	PLZ	Diagnose
Alice	76227	Schnupfen
Bob	76221	Husten
Carol	76221	Heiser
Dave	76229	Schnupfen
Alice	76227	Husten



encdb

enc	I_N	I_P	I_D
770E5B0F8	α	I	1
86D3EFBC2	β	II	2
EB51DA3C6	γ	II	3
15487E7B5	δ	III	1
8ABE19436	α	I	2

- Client

- entschlüsselt Zwischenergebnisse vom Server
- berechnet Endergebnisse

- Wie den Index kodieren? [1]**

Anfrageverarbeitung auf Buckets

■ Query Rewriting

- SELECT Diagnose FROM db
WHERE PLZ = 76221
- SELECT enc FROM encdb
WHERE $I_p = 'II'$

■ Zwischenergebnis vom Server entschlüsseln

- 86D3EFBC2 → {Bob, 76221, Husten}
- EB51DA3C6 → {Carol, 76221, Heiser}

■ Anfrageergebnis berechnen

- SELECT Diagnose FROM db
WHERE PLZ = 76221

Husten

Heiser

db

Name	PLZ	Diagnose
Alice	76227	Schnupfen
Bob	76221	Husten
Carol	76221	Heiser
Dave	76229	Schnupfen
Alice	76227	Husten



encdb

enc	I_N	I_P	I_D
770E5B0F8	α	I	1
86D3EFBC2	β	II	2
EB51DA3C6	γ	II	3
15487E7B5	δ	III	1
8ABE19436	α	I	2

Indexerstellung

- Kodierte Attribute als Indexeinträge
- Kodierung bzw. Verschlüsselung hat Einfluß auf Query-Performanz Sicherheit

- Kardinalität des Mappings
 - 1:1-Mapping von Attributwerten auf Index
 - N:M-Mapping von Attributwerten auf Index (mit $N \geq M$, „Bucketization“)

- Art der Kodierung
 - Zufällige Kodierung
 - Ordnungserhaltende Kodierung

db

Name	PLZ	Diagnose
Alice	76227	Schnupfen
Bob	76221	Husten
Carol	76221	Heiser
Dave	76229	Schnupfen
Alice	76227	Husten



encdb

enc	I_N	I_P	I_D
770E5B0F8	α	I	1
86D3EFBC2	β	II	2
EB51DA3C6	γ	II	3
15487E7B5	δ	III	1
8ABE19436	α	I	2

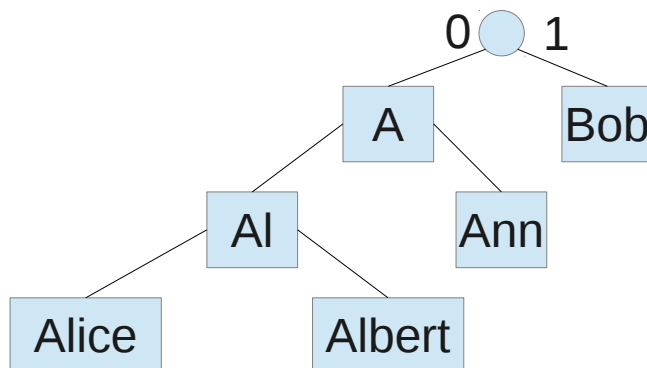
Art der Kodierung

Zufällig

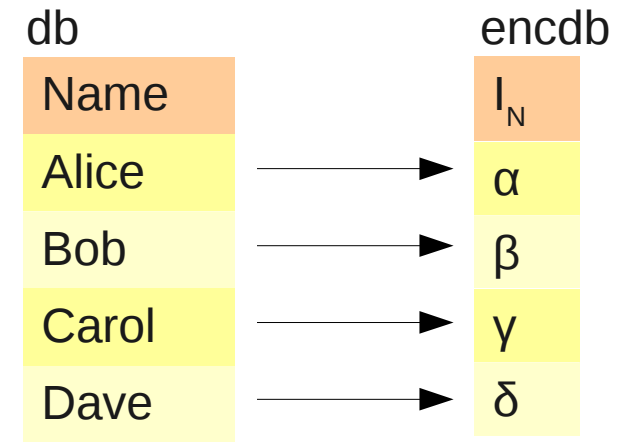
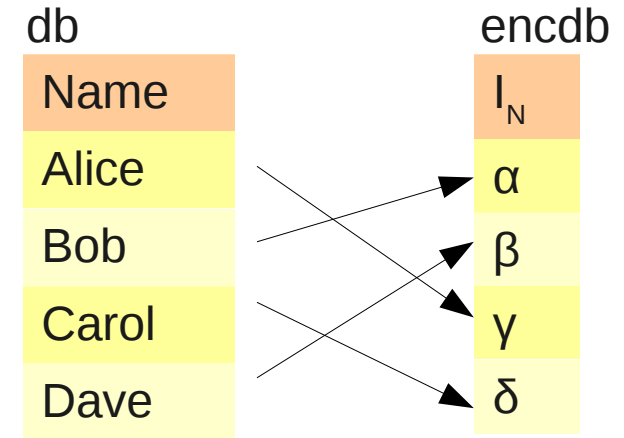
- Keine Ordnungsrelation zwischen Attributwerten und Indexwerten
- z.B. kryptographische Hashfunktion oder symmetrische Verschlüsselung

Ordnungserhaltend

- Bestehende Ordnungsrelation
- z.B. Prefix eines Tries bestimmt Hash-Bucket
- Performante Bereichsanfragen
- evtl. Zuordnungsangriffe möglich

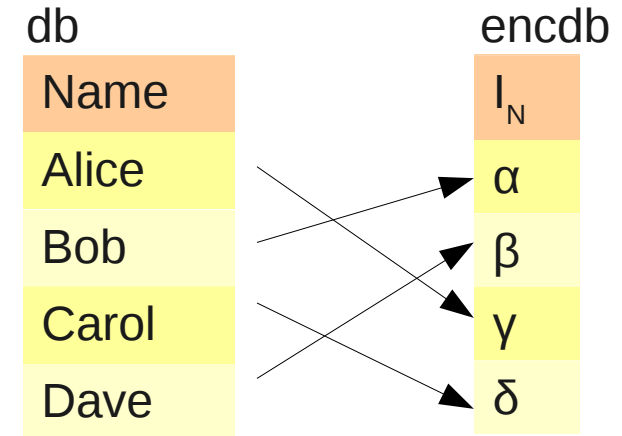


Anm.: Kodierung ist unabhängig davon, ob 1:1 oder M:N-Mapping

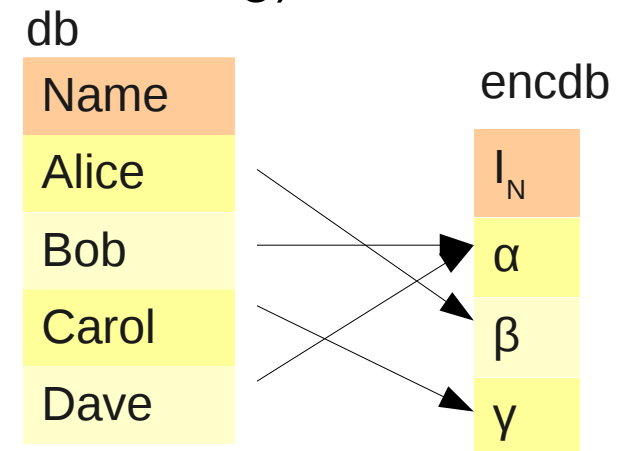


Mapping der Attributwerte

- 1:1-Mapping von Attributwerten auf Index
(ein Attributwert entspricht einem verschlüsselten Indexeintrag)
 - Symmetrische oder asymm. Verschlüsselung
 - Performante Exact-Match-Anfragen möglich
 - Frequenz- und Zuordnungsangriffe möglich



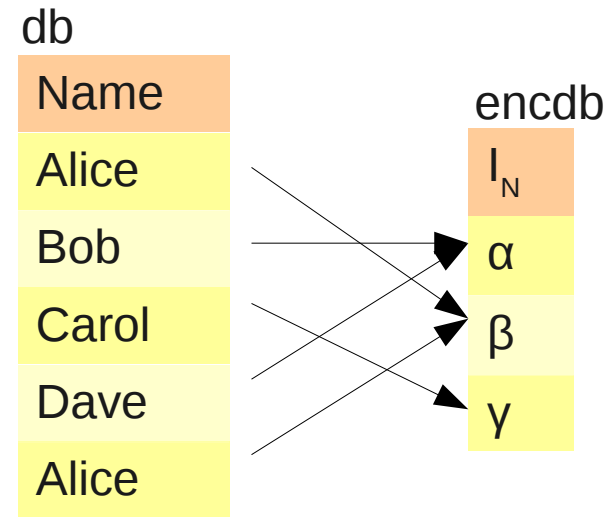
- Hash-Funktion über Attributwerte
(mehrere Attributwerte entsprechen dem selben Indexeintrag)
 - Konzept äquivalent zu Hash-Index
 - N:M-Mapping von Attributwerten auf Index
($N \geq M$, „**Bucketization**“)
 - Performanz, Frequenz- und Zuordnungsangriffe hängen stark von dem Mapping ab
→ **False Positives** bei der Anfrage



Die Hashfunktion

- Eigenschaften einer „sicheren“ Hashfunktion
 - Umkehrung ist nicht zu berechnen
 - nicht von Index auf Attributwert schließen
 - Attribute mit Gleichverteilung abbilden
 - Verteilung der Attributwerte auf Server entspricht nicht mehr Verteilung der Werte in den Hash-Buckets
 - Benachbarte Attributwerte landen nicht in benachbarten Buckets
 - Keine Ordnungsrelation zwischen Attributwerten und Indexwerten (kann für ordnungserhaltende Kodierung anders gelöst sein)

- Angreifbarkeit mit Hintergrundwissen?
 - nächste Folie



Angreifbarkeit Bucketization

1.) Wissen: encdb + Erwartungswerte der Attributverteilung in db
 z.B. Angreifer kennt Häufigkeit der Diagnosen

- Angriff auf Regel *Attributwert* → *Indexwert*
- Angriff zur Aufdeckung des Klartexts von db

Freq _{Name}		I _N
2	?	2α
1		2β
1		1γ
1		

2.) Wissen: encdb + db

z.B. Dienstleister wechselt von unverschlüsselter auf verschlüsselte DB

- Angriff auf Regel *Attributwert* → *Indexwert*

■ Messung der Exposure (Preisgabe):

- Zahl der möglichen Lösungen für die Abbildung von Attributwerten auf Indexwerte
 - Wenn es nur eine Lösung gibt, ist Schema untauglich
- Lösung für 2.): Alice muss α oder β sein; die 3 anderen permutieren: 6 Lösungsvarianten

Name		I _N
Alice	?	2α
Bob		2β
Carol		1γ
Dave		
Alice		

Query Performance

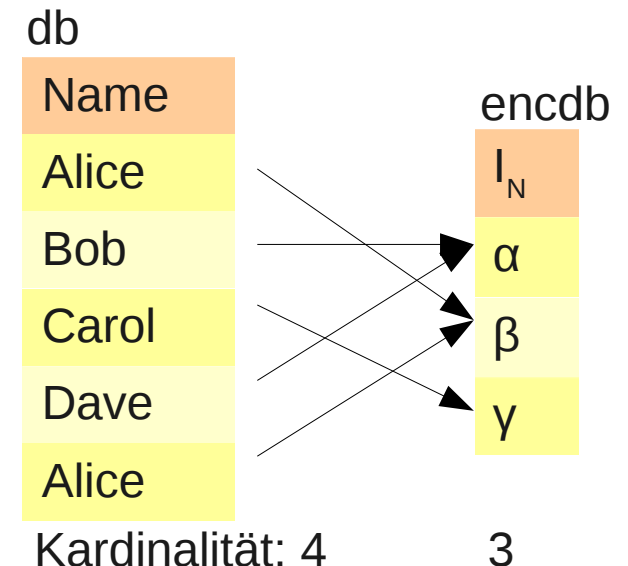
- Kardinalität der Attributwerte zur Kardinalität der Co-Domain der Hash-Funktion bestimmt Performanz von Exact-Match-Anfragen

- Beispiel

- `SELECT * FROM db WHERE Name = 'Bob'`
 - Rewriting:
`SELECT * FROM encdb WHERE $I_N = '\alpha'$`
 - Ergebnismenge: {Bob, Dave}
 - Client muss false positives entschlüsseln und filtern
 - Große Buckets: hoher Aufwand

- Was ist mit

- Bereichsanfragen?
 - Verbundanfragen wie Equi-Join, Theta-Join?
 - AVG, SUM, GROUP-BY, ORDER-BY?



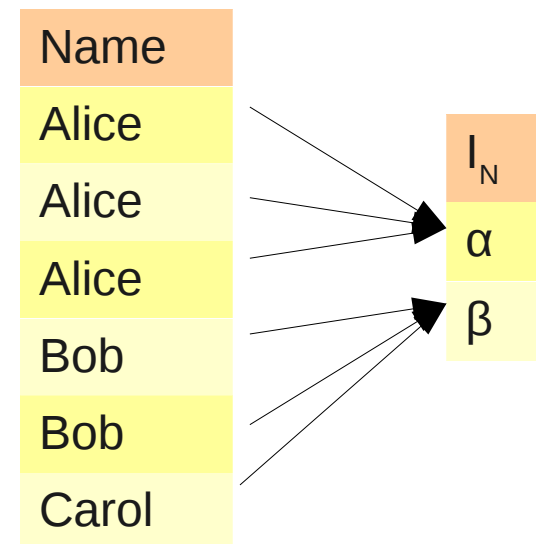
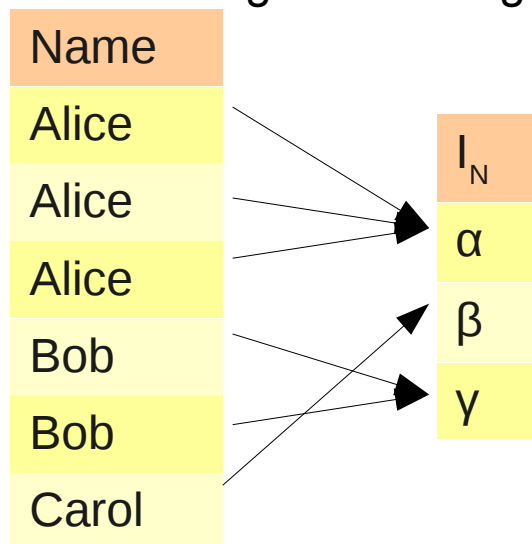
Diskussion Bucketization

■ Vorteile

- Kompromiss zwischen Performanz und Privatheit steuerbar
- Einfach einzusetzen, keine großen Änderungen an existierenden DBMS

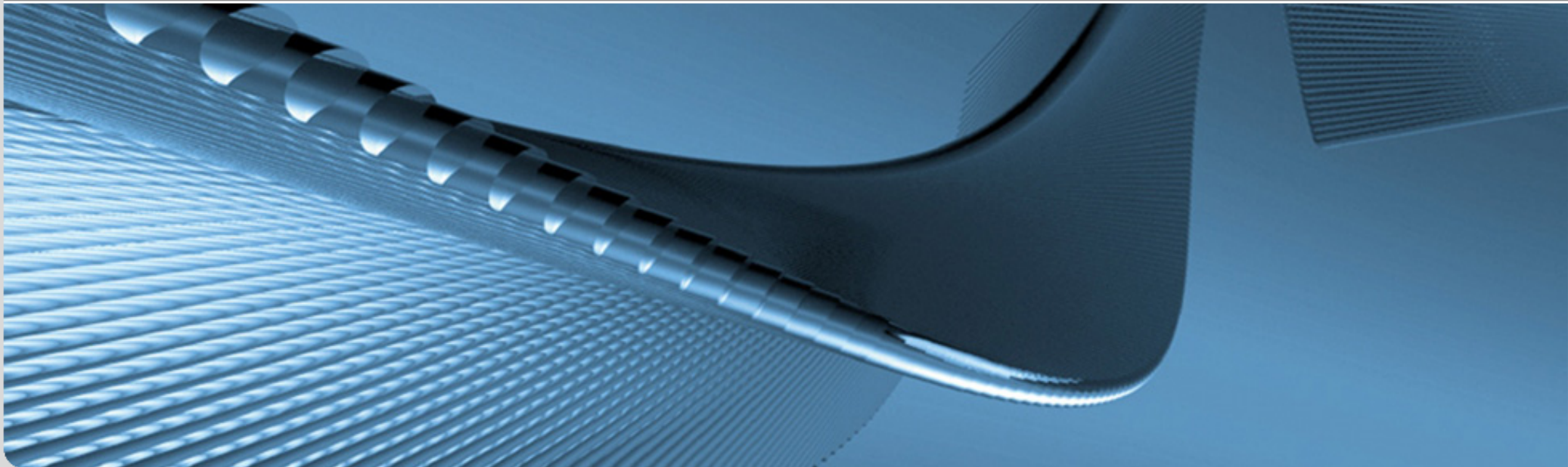
■ Nachteile

- Je nach Verteilung der Attributwerte (z.B. Power-Law)
 - Angreifbar durch einzigartige Verteilung der Indexwerte (1:1-Mapping) oder ordnungserhaltende Kodierung
 - Schlechte Performanz durch viele „false positives“ in den Buckets oder zufällige Kodierung



Mehrdimensionale Bucketization

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



Multidimensionale Bereichsanfragen

- Bereichsanfragen über mehrere (viele) Attribute

- SELECT Name FROM db
WHERE PLZ > 76225 AND PLZ < 76228
AND Gbt > 1982 AND Gbt < 1986

- Bisher betrachtet

- Fragmentation:

- Ganzes Fragment herunterladen und entschlüsseln, wenn Attribute nicht alle im Klartext im selben Fragment enthalten

- Bucketization

- viele false Positives
→ *schlaueres Verfahren?* [3]

			db
Name	Gbt	PLZ	Diagnose
Alice	1981	76227	Schnupfen
Bob	1974	76221	Husten
Carol	1995	76224	Heiser
Dave	2001	76229	Schnupfen
Eve	1985	76227	Husten
Frank	1978	76229	Heiser
George	1983	76226	Heiser
Harold	1999	76223	Husten

Schutzziel

■ Angreifermodell

- Angreifer ist **Honest-but-Curious**
- Angreifer hat Zugang zur Datenbank (Outsourcing-Szenario)

■ Angreifer

- Darf einzelne Werte in DB einfügen und Verschlüsselung beobachten (Chosen-Plaintext-Angriff)
- Will Informationen über die gespeicherten Daten bekommen
 - Ungefähre Verteilung der Werte
 - Näherungswerte für bekannte Individuen

Name	Gbt	PLZ	Diagnose
Alice	1981	76227	Schnupfen
Bob	1974	76221	Husten
Carol	1995	76224	Heiser
Dave	2001	76229	Schnupfen
Eve	1985	76227	Husten
Frank	1978	76229	Heiser
George	1983	76226	Heiser
Harold	1999	76223	Husten

Multidimensionale Bucketization

Client speichert

- Schlüssel
- Zuordnungsregel

α $\langle(1970,76220),(1980,76229)\rangle$

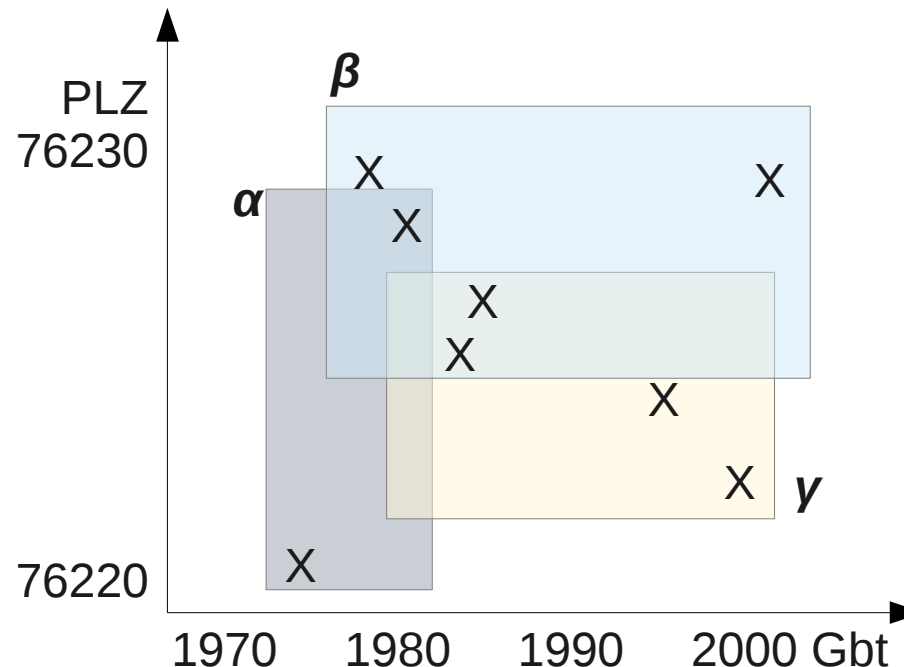
β $\langle(1975,76226),(2010,76233)\rangle$

γ $\langle(1980,76223),(2000,76227)\rangle$

Server speichert

- Datenbank,
tupelweise verschlüsselt
 $enc = encode(\{A_0, \dots, A_n\})$
- Bucket-ID (B)

Gbt	PLZ
1981	76227
1974	76221
1995	76224
2001	76229
1985	76227
1978	76229
1983	76226
1999	76223



enc	B
770E5B0F8	β
86D3EFBC2	α
EB51DA3C6	γ
15487E7B5	β
8ABE19436	γ
C2AB7DA00	α
2814F82C4	β
9B7DFAF00	γ

Anfrageverarbeitung

■ Query Rewriting

- SELECT * FROM db WHERE
PLZ > 76225 AND PLZ < 76228 AND
Gbt > 1982 AND Gbt < 1986
- SELECT enc FROM encdb WHERE B IN (β, γ)

α	<(1970,76220),(1980,76229)>
β	<(1975,76226),(2010,76233)>
γ	<(1980,76223),(2000,76227)>

■ Zwischenergebnis vom Server entschlüsseln

- {770E5B0F8, EB51DA3C6, 15487E7B5,
8ABE19436, 2814F82C4, 9B7DFAF0}
- {Alice, ...}, {Carol, ...}, {Dave, ...}
{Eve, ...}, {George, ...}, {Harold, ...}

enc	B
770E5B0F8	β
86D3EFBC2	α
EB51DA3C6	γ
15487E7B5	β
8ABE19436	γ
C2AB7DA00	α
2814F82C4	β
9B7DFAF0	γ

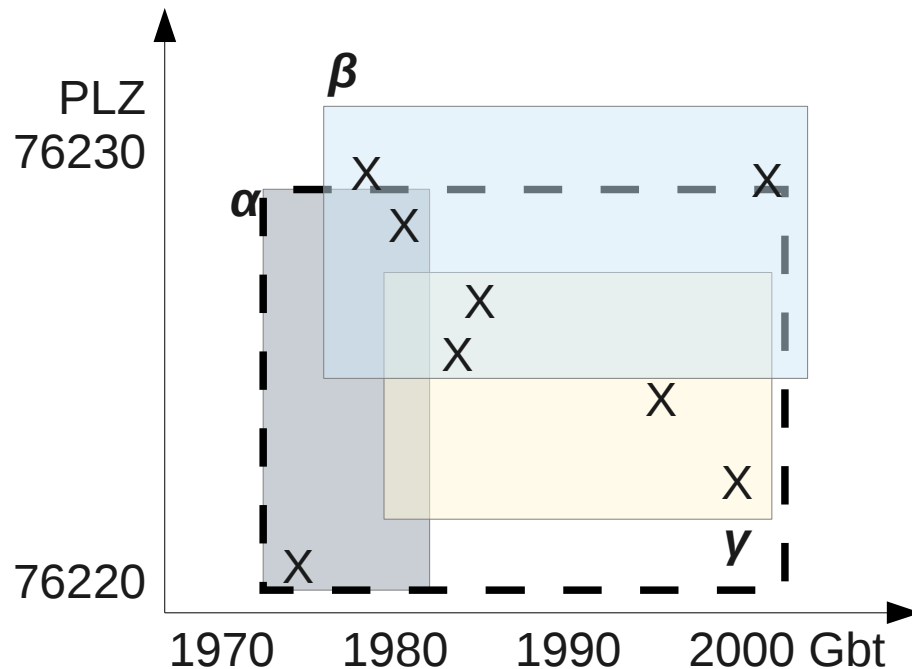
■ Anfrageergebnis berechnen

- false Positives herausfiltern

Eve	1985	76227	Husten
George	1983	76226	Heiser

Enthüllungsrisiko

- Angreifer kennt DB, darf Daten einfügen
 - Welches Bucket Label bekommt neuer Datensatz?
- Angreifer will Wertverteilung und/oder Näherungswerte auslesen
- Ausprobieren von Bucket Labels
 - MIN/MAX der Werte im Bucket werden offenbar
 - Jedes neue Label verrät detailliertere Informationen



enc	B
770E5B0F8	β
86D3EFBC2	α
EB51DA3C6	α
15487E7B5	β
8ABE19436	α
C2AB7DA00	α
2814F82C4	β
9B7DFafa0	α

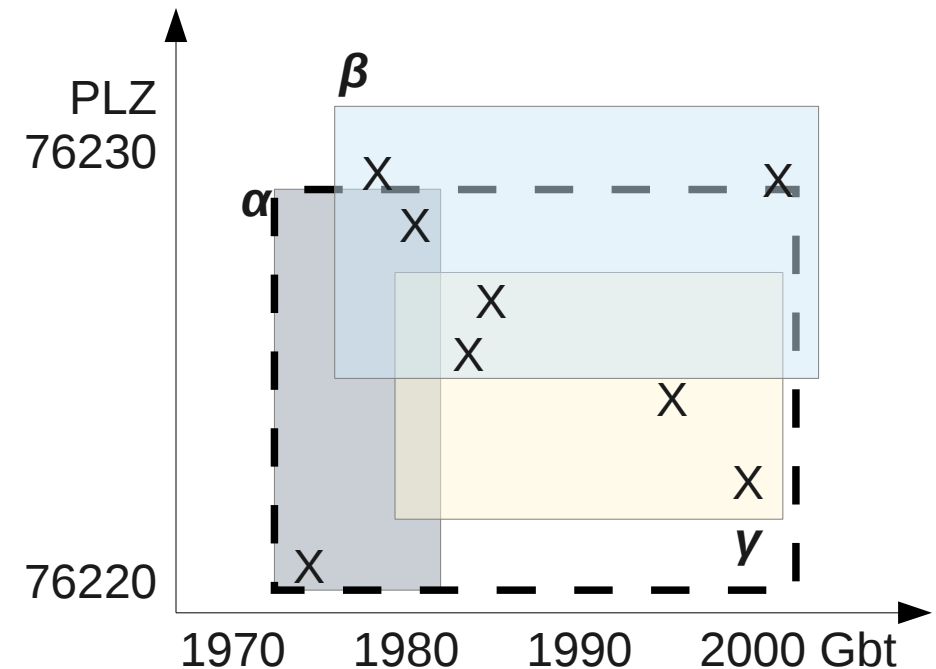


enc	B
770E5B0F8	β
86D3EFBC2	α
EB51DA3C6	γ
15487E7B5	β
8ABE19436	γ
C2AB7DA00	α
2814F82C4	β
9B7DFafa0	γ

Enthüllungsrisiko, formal

- „Wie gut kann ein Angreifer einen Attributwert aus dem Bucket Label vorhersagen?“
- Maß basiert auf der Entropie eines Buckets
 - Information einer Variable X über eine Variable Y:

$$H(X|Y) = - \sum_{x \in X, y \in Y} p(x,y) \log p(x|y)$$
 - *Anmerkung 1: Ziel ist das Maximieren der Entropie eines Buckets!*
 - *Anmerkung 2: Maß äquivalent zur I-Diversity!*



Optimale Verteilung der Buckets

- Extrem 1: Alle Werte in einem Bucket
 - Hohes Maß an Privatheit, keinerlei Rückschlüsse auf Attributverteilung
 - Maximale Anzahl von False Positives, schlechtestmögliche Performanz

- Extrem 2: Alle Werte in einem eigenen Bucket
 - Niedrigstes Maß an Privatheit, Buckets geben Attributverteilung wieder
 - Keine False Positives, aber riesengroße Anfragen beim Query Rewriting

- Was ist das richtige Maß?
 - 1) Zielfunktion für ein Optimierungsverfahren
 - wenn alle Bereichsanfragen gleich wahrscheinlich, M Buckets, jedes Bucket t mit F_t Datensätzen und Kantenlänge $b_{t,d}$ in Dimension $d \in D$:

$$\text{cost} = \sum_{t=1..M} (F_t^* \sum_{d=1..D} b_{t,d})$$
 - 2) Enthüllungsrisiko von vorangegangenen Folien
 - Maß für Risiko der Aufdeckung von sensiblen Daten

Algorithmus zur Bucket-Einteilung

- geg.: Datenbank db
Anzahl Buckets M
Performance degradation factor K
→ max. K mal mehr false Positives als mit optimaler Bucketaufteilung

- Algorithmus in 2 Schritten:

1. Query-Optimal Bucketization

starte mit einem Bucket mit allen Datensätzen in db

for j = 1 **to** M-1

 suche 2 Datensätze als neue Bucket-Grenzen, die die Kosten minimieren

 sortiere alle passenden Datensätze in neues Bucket ein

 berechne neue Bucket-Grenzen als Minimum Bounding Rectangles

end for

2. Controlled Diffusion

for each Bucket B

 verteile $K * |B| * M / |db|$ Datensätze zufällig auf andere Buckets

 berechne neue Bucket-Grenzen als Minimum Bounding Rectangles

end for

return M buckets

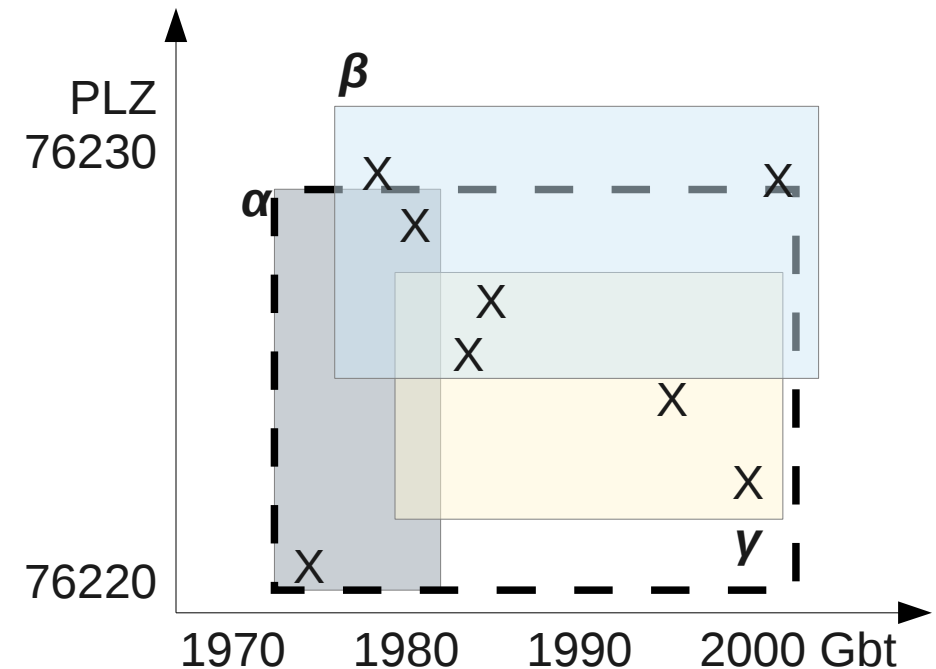
Diskussion multidim. Bucketization

■ Vorteile

- Verfahren zum Einstellen eines Kompromisses aus
 - Performanz (false positives) und
 - Datenschutz (Entropie der Buckets)

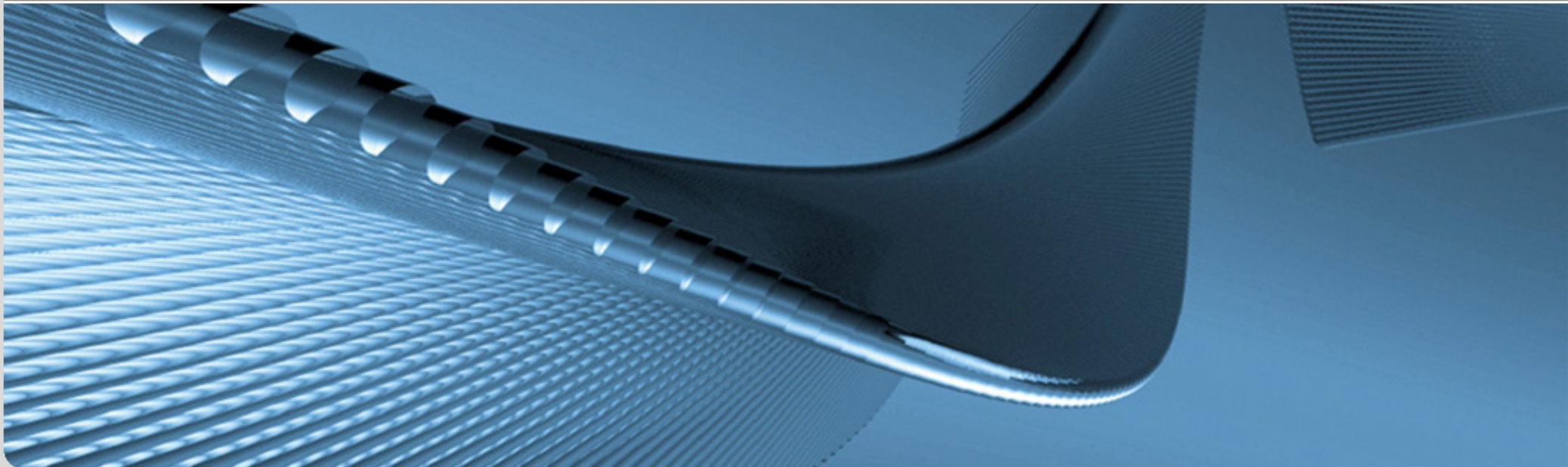
■ Nachteile

- Nur Greedy-Lösung für Bucket-Einteilung
 - Ggf. suboptimale Lösung
 - Abhängig von den Startparametern
- Nur performant bei mehrdimensionalen Bereichsanfragen
 - Bei anderen Anfragen viele false positives



Abschluss

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



Zusammenfassung

- Es gibt mehrere Verfahren zur Anfrageverarbeitung auf verschlüsselten Datenbanken

- Verfahren unterscheiden sich deutlich hinsichtlich
 - Performanz bei der Verarbeitung von unterschiedlichen Anfragen
 - manche Anfragearten erfordern komplette Entschlüsselung der Datenbank beim Client
 - Unterstützte Datenschutzziele

- In dieser Vorlesung
 - Fragmentation
 - schützt nicht gegen Ausspähen der Datenverteilung
 - sehr performant bei Anfragen auf sichtbare Attribute
 - (mehrdimensionale) Bucketization
 - flexibler Kompromiss aus Performanz (false positives) und Datenschutz

Literatur

- [1] Alberto Ceselli et al.: *Modeling and Assessing Inference Exposure in Encrypted Databases*. In: ACM Transactions on Information and System Security, 8(1), 2005
- [2] Valentina Ciriani et al.: *Combining Fragmentation and Encryption to Protect Privacy in Data Storage*. In: ACM Transactions on Information and System Security, 13(3), 2010
- [3] Bijit Hore et al.: *Secure Multidimensional Range Queries over Outsourced Data*. In: VLDB Journal 21/2012