February 2005

# Effizienter unbeobachtbarer Datenbankzugriff

Oliver Berthold
*Humboldt-Universität zu Berlin*

In: Ferstl, Otto K, u.a. (Hg) 2005. *Wirtschaftsinformatik 2005: eEconomy, eGovernment, eSociety*; 7. Internationale Tagung Wirtschaftsinformatik 2005. Heidelberg: Physica-Verlag

ISBN: 3-7908-1574-8

# Effizienter unbeobachtbarer Datenbankzugriff

**Oliver Berthold**
Humboldt-Universität zu Berlin

*Zusammenfassung: Es wird ein effizentes Verfahren zum unbeobachtbaren, privaten Datenbankzugriff (PDA) vorgestellt. Die Server-Komponente des Systems besteht aus einem oder mehreren Sicheren Coprozessoren (SC's), d.h. physisch geschützten Rechner-Modulen, die einen potentiell beobachtbaren Datenspeicher nutzen. Einem die SC's nicht kontrollierenden, ansonsten aber omnipräsenten Angreifer bleibt verborgen, welche Information bei einer Nutzeranfrage abgefragt werden. Im Unterschied zu üblichen „PDA"-Ansätzen, die meist auf Vertrauensverteilung und Kryptographie basieren, kann mit dem vorgestellten PDA-Schema eine höhere Effizienz und praktische Einsatzfähigkeit erzielt werden. Zudem ermöglicht die Verwendung von SC's breitere Einsatzgebiete i.d.R ohne spezielle Software im Nutzerrechner, z.B. anonyme Web(service)-, Datenbank-, File-, Community- und Payment-Dienste.*

*Schlüsselworte: Private Information Retrieval, Privacy Enhanging Technologies, Unbeobachtbarkeit, Anonymität*

## 1 Einführung

In diesem Artikel wird ein Verfahren vorgestellt, welches eine unbeobachtbare Nutzung von Diensten und den Abruf von Informationen über öffentliche Datennetze wie das Internet ermöglicht. Dazu wird ein „Datenbank"-System verwendet, welches Informationen und verschiedene Dienste anbieten kann. Einzig beobachtbar ist dabei, dass ein Zugriff auf die Datenbank erfolgt. Unbeobachtbar ist, welcher Dienst genutzt bzw. welche Information abgerufen wird. Insbesondere soll auch der Serviceanbieter selbst dies nicht erfahren.

Derartige Private Information Retrieval (PIR)-Techniken stellen eine Methode dar, personenbezogene Nutzerdaten vor Missbrauch zu schützen. Wie z.B. in [PIP00] gezeigt, werden derartige Missbrauchsmöglichkeiten als wesentliche Hinderungsgründe für die Nutzung von Online- und Internetdienstleistungen angegeben.

Absolute Unbeobachtbarkeit gegenüber einem allmächtigen Angreifer ist nicht möglich. So wird grundsätzlich unterstellt, dass der Rechner des Nutzers nicht vom Angreifer kontrolliert wird. Da dem Nutzer die Informationen „im Klartext“ dargestellt werden müssen, wäre eine Überwachung spätestens durch Beobachten der Ausgabegeräte möglich. Alle in Abschnitt 2 vorgestellten bisher bekannten Verfahren schränken die Stärke des Angreifers zusätzlich ein.

In dem hier vorgestellten Verfahren wird unterstellt, dass der Angreifer auf bestimmte Geräte keinen Zugriff hat. Ein Beispiel für ein solches Gerät ist ein sicherer Coprozessor (SC), z.B. IBM Serie 4758 (http://www.ibm.com/security/cryptocards/). Dies ist ein in PC's integrierbares Modul, das selbst einen vollwertigen PC darstellt und sich in einem physisch geschützten und mit Manipulationsensoren ausgestatteten Gehäuse befindet.

Vertraut man IBM bezüglich der aufgeführten Eigenschaften, kann der SC unter Verwendung digitaler Signaturen Dritten seine Unversehrtheit beweisen[1] und angeben, welche signierten Programme bzw. Betriebssysteme in welcher Konfiguration im Modul laufen. Wird das Programm überprüft – beispielsweise durch eine Veröffentlichung als Open-Source – ist der Angreifer weder physisch noch logisch in der Lage, den SC zu kontrollieren.

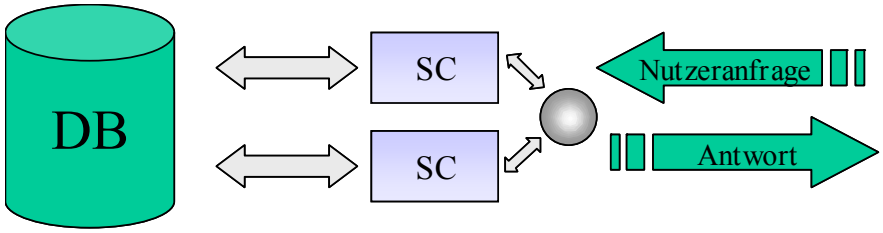


Abbildung 1: Aufbau des PDA-Schemas

Da derartige SC’s nur über beschränkte Ressourcen verfügen, müssen die eigentlichen Daten in externen Datenspeichern abgelegt werden. Das vorgestellte Verfahren schlägt einen Algorithmus vor, der eine effiziente Nutzung dieses externen und daher prinzipiell beobachbaren Datenspeichers ermöglicht. Das zu lösende Problem ist, einem Beobachter der Zugriffe auf den Datenspeicher möglichst keine Rückschlüsse über die Anfrage zu erlauben, gleichzeitig aber nicht bei jeder Anfrage einmal die gesamte Datenbank durchsuchen zu müssen. Abbildung 1

[1] Solange das Gehäuse unbeschädigt ist, kann der SC bestimmte Dokumente wie z.B. die Softwarekonfiguration signieren, wobei der Signierschlüssel von IBM zertifiziert ist.

zeigt den Aufbau eines PDA-Systems (Private Database Access). Es besteht aus einer Datenbank, welche über $N$ Datensätze der Länge $l$ verfügt. Die technische Realisierung dieses Speichers ist beliebig. Zudem besteht das System aus einem oder mehreren SC's, die die Möglichkeit haben, auf beliebige Datensätze der Datenbank zuzugreifen und untereinander und mit den Nutzerrechnern zu kommunizieren. In dem hier beschriebenen Schema sind die Datensätze verschlüsselt und durchmischt in der Datenbank gespeichert. Die Zuordnung zwischen der Datensatznummer $x$ und der Speicherposition $SP(x)$ ist nur dem/n SC's bekannt. In einem initialen Schritt muss daher die gemischte Datenbank aus einer Originaldatenbank erstellt werden.

Ziel der PDA-Verfahren ist es, bei einer Anfrage keine Rückschlüsse über den angefragten Datensatz aus der Beobachtung der Zugriffe auf Speicherpositionen $SP$ zu erlauben.

In den folgenden Abschnitten werden PDA bzw. PIR-Verfahren von Anonymisierungsverfahren abgegrenzt und mögliche Einsatzgebiete aufgezeigt. In Abschnitt 2 werden andere PIR-Verfahren vorgestellt, insbesondere auch bisherige Techniken, die auf der Nutzung von SC's basieren. In Abschnitt 3 wird das vorgeschlagene Verfahren unter Nutzung eines einzelnen SC beschrieben. Darauf folgt im Abschnitt 4 eine Diskussion zu Fragen der Parallelisierungsmöglichkeiten und der Datensicherheit. In Abschnitt 5 werden die Aussagen das Artikels zusammengefasst und Sicherheit und Effizienz des Verfahrens mit anderen PIR-Verfahren verglichen.

## 1.1 Anonymisierungsdienste

Anonymisierungsdienste wie z.B. JAP[2] ermöglichen üblicherweise den Zugriff auf beliebige Informationen im Internet[3]. Anon-Dienste sind daher flexibel einsetzbar und unabhängig von den angebotenen Informationen [BFKö01]. Dies bedeutet jedoch auch, dass ein Inhalteanbieter Anonymität nicht (allein) anbieten kann, da dazu immer ein zwischengeschalteter unabhängiger Dienst existieren muss, der anstelle des Nutzers dessen Anfragen zum Inhalteanbieter sendet und so den eigentlichen Sender anonymisiert, i.e. transparent erscheinen lässt. Unbeobachtbarkeit mit Hilfe von PDA-Datenbanken kann direkt angeboten werden, sogar ohne dass der Nutzer spezielle Software installieren muss, wie dies bei Anon-Diensten, die gegen stärkere Angreifer schützen, unumgänglich ist.

Zudem anonymisieren derartige Dienste nicht perfekt, da absolute Anonymisierung inakzeptabel hohe Reccourcen benötigen würde. Insbesondere problematisch

---

2 JAP Anonymity & Privacy; http://www.anon-online.de

3 Anon-Dienste anonymisieren den einzelnen Nutzer in einer Gruppe von Nutzern, die sich gleichartig verhalten, also i.d.R. gerade den gleichen Anon-Dienst nutzen. Die Informationsabrufe sind dagegen i.d.R. beobachtbar.

sind langzeitig verkettbare Aktionen, wie z.B. der regelmäßige Abruf eines persönlichen Postfaches. Da üblicherweise nur eine bestimmte Person Zugriff auf ein Postfach hat, muss jeder beobachtete Zugriff von dieser Person stammen. Da der Dienst aber zu verschiedenen Zeitpunkten unterschiedliche Nutzergruppen hat, kann ein globaler Beobachter Schittmengen bilden: Nur der Besitzer des Postfaches ist zu allen beobachteten Zeitpunkten online, die Anonymität sinkt daher mit jeder Nutzung [BFKö01].

## 1.2 Einsatzgebiete/Dienstespektrum

Bei allen bisher bekannten PIR-Verfahren sind einzig Anfragen vom Typ „Gib mir Datensatz $\boldsymbol{x}$“ möglich. Zudem haben alle Datensätze eine einheitliche Länge $l$, um Rückschlüsse anhand der Länge einer beobachteten Antwort zu verhindern.

Trotz dieser starken Einschränkung sind prinzipiell beliebige Dienste auf einer solchen Datenbank realisierbar. Beispielsweise kann durchaus eine SQL-Datenbank realisiert werden, wobei das PIR-Verfahren den physischen Zugriff auf einen blockorientierten Sekundärspeicher realisiert. Das Datenbankmanagement muss in eine sichere Umgebung (Client oder ein SC), verlagert werden.

Die in Aussicht gestellte breitere Einsatzmöglichkeit des in diesem Artikel vorgeschlagenen Verfahrens begründet sich durch folgende Eigenschaften:

1. Die Verwendung eines sicheren Gerätes ermöglicht die Unterstützung nahezu beliebiger Protokolle/Dienste, solange eine Verschlüsselung der Verbindung vorgesehen ist. Insbesondere ist keine spezielles Clientsoftware wie bei anderen Lösungsansätzen erforderlich. Beispiele für problemlos realisierbare Protokolle wären: Secure FTP (Download), POP3 SSL (E-Mail-Empfang), Web (HTTPS) und Webservices.
2. Das Verfahren ermöglicht unbeobachtbares Ändern von Daten der Datenbank (Abschnitt 3), so dass interaktive Dienste realisierbar sind. Alle anderen bekannten PIR-Verfahren (Abschnitt 2) ermöglichen nur das unbeobachtbare Lesen eines festen Datenbestandes.

Im Folgenden werden beispielhaft einige mögliche Anwendungen aufgelistet:

**Webangebote:** Beratungsdienste, Patentdatenbanken, Jobagenturen etc. könnten ihren Kunden eine unbeobachtbare Nutzung ihrer Webseiten anbieten. Das SC oder eine spezielle Nutzersoftware muss dabei die Verwaltung der Datensätze und die Zuordung zu Dateien übernehmen. Im ersten Fall braucht der Nutzerrechner keine spezielle Software. Es genügt ein gewöhnlicher Webbrowser. Der Zugriff erfolgt über eine verschlüsselte (HTTPS) Verbindung.

**Webservices:** Immer mehr Informationen/Dienste werden per Webservice zur automatisierten Nutzung beispielsweise durch ansonsten lokal laufende Programme bereitgestellt. Zukünftig werden daher vermehrt Programme nicht mehr nur lokal,

sondern teilweise „im Internet“ ausgeführt. Die dabei unbemerkt weitergegebenen Daten können sehr sensibel sein. PDA-Webservices können einen Beitrag zur Lösung dieser Problematik leisten, insbesondere für hauptsächlich zur Bereitstellung von Informationen (Daten-Services) konzipierte Webservices.

**Zugangsbeschränkte Dienste:** Da Nutzer eines PDA-Services nicht notwendigerweise anonym sein müssen, ist die Überprüfung von Zugangsberechtigungen problemlos möglich. Aufgrund der Unbeobachtbarkeit der konkreten Anfragen wird jedoch zuverlässig verborgen, auf welche Daten ein Nutzer zugegriffen hat. Durch individuelle Verschlüsselung der *DS* oder Nutzung von SC’s kann eine Zugriffsbeschränkung bestimmter Daten realisiert werden. Die beiden folgenden Anwendungen setzen eine solche Zugriffsbeschränkung voraus.

**Mail/Communitydienste:** In einer PDA-Datenbank können persönliche u.a. per Mailprogramm (POP3 SSL) abrufbare Postfächer gespeichert werden. Wie im letzten Abschnitt argumentiert, sind gerade langzeitig verkettbare Aktionen wie das Abrufen eines persönlichen Postfaches nur ungenügend durch Anonymisierungsdienste zu schützen. Für das anonyme Senden an ein solches Postfach benötigt man ein PDA-System, welches unbeobachtbare Änderungen von Daten unterstützt, was bisher nur das hier vorgeschlagene Verfahren unterstützt. Jedoch muss für das Senden von Nachrichten die Anonymitätsgruppe (die Gruppe der möglichen Sender) berücksichtigt werden. Eine praktikable Lösung wäre ein zeitverzögerter Nachrichtenversand: Der Schreibzugriff zur PDA-Datenbank wird vom Empfang einer Nachricht zeitlich entkoppelt. Als Sender kommen somit alle Nutzer in Frage, welche die PDA-Datenbank in dem konfigurierten maximalen Verzögerungsinterval genutzt haben.

**Micro-Payment:** Da der Nutzer nicht anonym sein muss, kann problemlos eine Abrechnung nach abgefragter Informationsmenge (in Patentdatenbanken z.B. Anzahl abgerufener Patente) erfolgen. Durch Nutzung eines sicheren Gerätes sind jedoch auch anonyme Bezahlsysteme realisierbar. Konkret könnte die Datenbank pseudonyme Nutzerkonten verwalten und somit Nutzern eine unbeobachtbare Bezahlung interner (Dienste die von der PDA-Datenbank selbst angeboten werden) und externer Dienste ermöglichen. Die pseudonymen Konten können über übliche, nicht notwendigerweise anonyme Zahlungsmittel ge- bzw. entladen werden. Unbeobachtbar sind Transfers (Geldflüsse) zwischen den PDA-Konten. Ein derartiger Dienst benötigt wiederum einen PDA-Service, der wie das vorgestellte Schema unbeobachtbare Schreibzugriffe ermöglicht. Wiederum muss auf eine zeitverzögerte Zustellung der Zahlungseingänge geachtet werden.

Eine PDA-Datenbank kann prinzipiell mehrere verschiedene Dienste zugleich anbieten, wobei einem Beobachter verborgen bleibt, welcher konkrete Dienst genutzt wird. Einzig beobachtbar ist die Nutzung der PDA-Datenbank.

Zur Demonstration der möglichen Dienste wurde das beschriebene PDA-Schema prototypisch implementiert und steht online als Service unter http://www.pivacy-database.de zur Verfügung.

## 2 Bisherige Forschungsarbeiten

Abgegrenzt werden muss das hier bearbeitete Forschungsgebiet von Ansätzen wie [HILM02], bei welchen Datenbanken an Service-Provider ausgelagert werden, ohne diesem Daten bzw. Suchanfragen zu offenbaren. Bei derartigen Ansätzen werden die einzelnen Datensätze einer Relation verschlüsselt und über zusätzliche, ebenfalls verschlüsselte Indexe eine effiziente Suche auf der Relation ermöglicht. Der Hauptunterschied zu den hier betrachteten PIR-Ansätzen ist die Einschränkung, dass die ausgelagerte Datenbank nur für eine geschlossene und völlig vertrauenswürdige Nutzergruppe zugänglich sein darf, da jeder Nutzer den Schlüssel zum Entschlüsseln der vom Service-Provider empfangenen Datensätze kennen muss. Wenn man diese Aufgabe an einen SC ausgelagert, müsste der Schlüssel nicht jedem Nutzer bekannt sein, jedoch tritt dann das folgende Problem auf: Kann der Angreifer gezielte Anfragen an die Datenbank richten, so kann er durch Beobachten der Speicherzugriffe des Service-Providers verschlüsselte Datensätze identifizieren bzw. auf kleinere Gruppen eingrenzen und somit die Ziele späterer Anfragen identifizieren. Die PIR-Ansätze erlauben es, unbeobachtbar zugreifbare Datenbanken öffentlich zugänglich zu machen.

Das PIR-Problem wurde erstmals 1995 in [CGKS95] vorgestellt und seither wurden in über 30 Forschungspapieren Lösungen vorgeschlagen. Die Lösungsansätze lassen sich in drei Gruppen einteilen, abhängig davon, wie das Vertrauen in Dritte organisiert wird. Ein eher theoretisches Konzept vertraut ausschließlich auf die Sicherheit von kryptographischen Algorithmen. Es muss also keinem Dritten vertraut werden. Allerdings sind derartige Verfahren wie [KO97] oder [Chan04] nicht praktisch anwendbar, da einerseits die Datensatzlänge nur $l$=1 Bit beträgt und zudem pro Anfrage für jeden Datensatz (jedes Bit) der Datenbank eine aufwendige Rechenoperation, vergleichbar mit dem Aufwand für eine asymmetrische Verschlüsselung, durchgeführt werden muss.

Ein weiteres Konzept ist die Verteilung des Vertrauens auf mehrere Stationen, wobei es i.d.R. ausreicht, wenn eine der Stationen vertrauenswürdig ist, um die Unbeobachtbarkeit der Nutzeranfrage zu schützen.

Das dritte Konzept ist die Verwendung von physisch geschützten Geräten, die zwischen Nutzer und Datenbank geschaltet werden und die Nutzeranfragen von den Zugriffen auf die Datenbank entkoppeln sollen. Zu diesen beiden Ansätzen werden Beispielverfahren in den folgenden Abschnitten erläutert.

Ein trivaler Ansatz ist das Versenden der gesamten Datenbank an jeden Nutzer – der Kommunikationsaufwand zur Übermittlung eines Datensatzes ist $O(N)$. Bei diesem Ansatz muss ebenfalls keiner dritten Partei vertraut werden, mit Ausnahme des Nutzerrechners. Als reale Beispiele für diese Methode könnten Rundfunk- und Fernsehübertragungen angenommen werden: Alle Kanäle werden bis zu jedem Endgerät transportiert und erst dort lokal ein Kanal ausgewählt. Effiziente Verfahren sollten einen geringeren Kommunikationsaufwand aufweisen.

Idealerweise sollte der Berechnungsaufwand innerhalb des PIR/PDA-Servers möglichst gering sein. Der wesentliche Berechnungsaufwand ist dabei der Zugriff auf die Datensätze der Datenbank. Leider ist dieser Aufwand nicht beliebig minimierbar, wenn man absolute Unbeobachtbarkeit garantieren will: Soll jeder der $N$ Datensätze einer Datenbank mit der gleichen Wahrscheinlichkeit ($P$=1/$N$) angefragt worden sein, muss man pro Anfrage mindestens einmal auf jeden Datensatz zugreifen, da nichtangefragte Datensätze sonst ausgeschlossen sind. Bei einer Lösung mit mehreren Datenbanken muss dies für die Gesamtheit der Server gelten. Dieser Aufwand ist für viele praktische Anwendungen unrealistisch. Daher wird in Abschnitt 3 ein Verfahren vorgestellt, welches nur probabilistische Sicherheit bietet, dafür aber einen von der Datenbankgröße unabhängigen Berechnungsaufwand und einen optimalen Kommunikationsaufwand aufweist.

## 2.1 Verteiltes Vertrauen

Bei dem Konzept der Vertrauensverteilung geht man von mehreren Stationen mit jeweils replizierten, also identischen Datenbanken aus. Die Verfahren dieses Konzeptes sind sicher, sobald es mindestens eine der Stationen einschließlich der Datenspeicher vom Angreifer nicht kontrolliert werden kann.

Ein einfaches Verfahren wurde in [CoBi_95] vorgestellt. Dabei sendet der Nutzer an jede der $M$ Server verschlüsselt einen Bit-Vektor der Länge $N$, wobei jedes Bit einen Datensatz der Datenbank adressiert. Die Aufgabe jedes Servers ist es, alle Datensätze, die im Vektor mit einer „1“ markiert sind, miteinander bitweise zu überlagern (logische XOR-Verknüpfung) und das Ergebnis zum Nutzer zu senden. Die Anfragevektoren bildet der Nutzer, indem er $M$-1 Vektoren zufällig bildet. Der $M$-te Vektor ergibt sich aus der bitweisen Überlagerung der anderen $M$-1 Vektoren, wobei die Bitstelle des abzufragenden Datensatzes umgekippt wird. Die erhaltenen $M$ Antworten ergeben bitweise überlagert den gewünschten Datensatz, da durch die Überlagerung der Anfragevektoren alle Datensätze bis auf den angefragten in geradzahliger Anzahl abgerufen werden und sich damit im Ergebnis auslöschen. Das Verfahren ist sicher, da nur mit Kenntnis aller $M$ Anfragen bzw. Serverantworten der angefragte Datensatz ermittelt werden kann.

In [BCKP01] wird eine Optimierung des Kommunikationsaufwandes des Verfahrens vorgestellt. Allerdings beträgt dieser trotzdem $O(N)$, da in der Anfrage für jeden Datensatz mindestens ein Bit übertragen werden muss.

Weitere Verfahren die auf Datensätzen mit $l$>1 arbeiten sind [CGKS95; CGN97; Gil00]. Alle Verfahren versuchen, den Kommunikationsaufwand unter $O(N)$ zu verringern. Bei [CGKS95] beträgt dieser jedoch für $M$=2 ca. 30% der Datenbankgröße. Zudem haben die Verfahren eine höherere Berechnungskomplexität, da pro Datensatz eine wesentlich kompliziertere Rechenoperation als die XOR-Verknüpfung ausgeführt werden muss. Mindestens 10 weitere Artikel beschreiben

Verfahren für Datensätze mit $l$=1, welche einen ähnlichen Aufwand aufweisen, allerdings pro 1 Bit-Datensatz.

## 2.2 Vertrauen in physisch geschützte Hardware

Bei diesem Konzept wird ein SC, ein physisch geschütztes Gerät verwendet. Dabei ist das Vertrauensmodell, dass der Angreifer dieses Gerät nicht kontrollieren, aber dessen Zugriffe auf die Datenbank beobachten kann.

In [SS00] wird eine Basismethode beschrieben, bei welcher der SC bei jeder Anfrage die gesamte Datenbank abruft, den gewünschten Datensatz zwischenspeichert und diesen danach an den Nutzer sendet. Optimierungen sehen den Einsatz mehrerer SC's und die Durchführung mehrerer paralleler Anfragen vor. Der Kommunikationsaufwand zum Nutzer ist optimal, d.h. unabhängig von der Datenbankgröße, der Berechnungsaufwand $O(N)$.

In [AsFr02] wird ein Verfahren vorgestellt, welches auf einer schon in Abschnitt 1 beschriebenen gemischten und verschlüsselten Datenbank beruht.

Eine Anfrage $Q_i$ nach DS $x$ wird durch Zugriff auf dessen Position $SP(x)$ und zusätzlich auf alle auf dieser Datenbank bisher abgerufenen Positionen $SP_{j<i}$ bearbeitet. Würde man nur auf $SP(x)$ zugreifen, erhielte der Beobachter Information über den zugegriffenen Datensatz. Beispielsweise könnte er selbst alle Datensätze abfragen und durch Beobachten der Datenbankzugriffe die Position jedes Datensatzes ermitteln. Die Ziele folgender Anfragen könnten dann durch Beobachten des DB-Zugriffs einfach ermittelt werden.

Der Aufwand pro Anfrage wächst somit linear mit der Anzahl der Anfragen. Um die Antwortzeit niedrig zu halten muss die Umsortierung der Datenbank nach ca. $\sqrt{N}$ Zugriffen wiederholt werden. Der Aufwand für eine perfekt unbeobachtbare Umsortierung liegt bei $O(N*\log N)$. Der Gesamtaufwand des Schemas liegt somit ebenfalls bei ca. $O(N)$ Datenbankzugriffen pro Anfrage, die sich jedoch nicht auf die einzelne Antwortzeit auswirken, da die Umsortierung separat erfolgt.

In [Ason03] stellt Asonov ein probabilistisches PDA-Verfahren vor. Die Idee ist den Berechnungsaufwand zu senken, indem unterschiedliche Anfrage-Wahrscheinlichkeiten für die einzelnen Datensätze der Datenbank zugelassen werden.

Bei diesem Protokoll erfolgen zur Bearbeitung einer Anfrage $a$ Zugriffe auf der unverschlüsselten Originaldatenbank und $b$ Zugriffe auf der umsortierten und verschlüsselten Datenbank, wobei beide Datenbanken inhaltlich identisch sind. Die Zugriffe auf die gemischte Datenbank erfolgen dabei einfach sequentiell von $SP_0$ bis $SP_N$ und die Zugriffe auf die Originaldatenbank mit jeweils $a$-1 Zugriffen auf zufällige Datensätze und einem Zugriff für den angefragten Datensatz, falls dieser nicht bereits in der Menge der Zugriffe aus der gemischten Datenbank enthalten

ist. Es existieren folglich zwei Gruppen von Datensätzen mit unterschiedlichen Anfragewahrscheinlichkeiten, die *a* Datensätze aus der Originaldatenbank und alle übrigen.

Die Sicherheit des Verfahrens basiert darauf, dass der Beobachter nicht erkennen kann, ob und wenn ja welcher der von der Originaldatenbank abgerufenen Datensätze der zum Nutzer gesendete ist. Da der Beobachter die von der verschlüsselten Datenbank abgerufenen Datensätze nicht identifizieren kann, ist zudem sichergestellt, dass jeder beliebige Datensatz ein mögliches Ziel der Nutzerfrage ist. Dies impliziert jedoch auch, dass die verschlüsselte Datenbank, nachdem sie vollständig abgefragt wurde, mit $O(N*\log N)$ neu gemischt werden muss, da der Beobachter sonst zusätzlich Informationen erhält. Minimalen Aufwand erzielt man mit a=1 und b=1. In diesem Fall ist jedoch der von der Originaldatenbank abgerufene Datensatz mit extrem hoher Wahrscheinlichkeit $P=(N-1)/N$ der vom Nutzer angefragte Datensatz. Trotzdem beträgt der Aufwand noch $O(\log N)$. Für eine ausgeglichene Wahrscheinlichkeitsverteilung, d.h. für perfekte Sicherheit, ist der Aufwand höher als $O(N)$.

# 3 Probabilistisches PDA-Schema

Wie im letzten Abschnitt gezeigt, bieten fast alle bisher bekannten PIR-Verfahren absolute Unbeobachtbarkeit. Der Preis dafür ist jedoch ein Aufwand in $O(N)$ zur Bearbeitung einer bzw. einer kleinen Anzahl simultaner Anfragen, infolge dessen sind diese Verfahren praktisch kaum einsetzbar. Es wird daher hier ein Verfahren vorgestellt, welches nur probabilistische Sicherheit bietet, dafür aber einen relativ geringen, von der Datenbankgröße unabhängigenen Berechnungs- und Kommunikationsaufwand von $O(1)$ aufweist. Ein solches Verfahren wäre für beliebig große Datenbanken effizient einsetzbar.

## 3.1 Algorithmus

Für die Beschreibung des Verfahrens werden folgende Formalismen vereinbart:

| | |
|---|---|
| ***DB*** | Gemischte Datenbank mit *N* Datensätzen (*DS*) der Länge *l*. |
| $\boldsymbol{Q_i(x)}$ | *i*'te Anfrage an das SC, wobei nach Datensatz *x* gefragt wird. |
| $\boldsymbol{SP_i(x)}$ | Physische Position von Datensatz *x* in der gemischten Datenbank zum Zeitpunkt *i*. |
| $\boldsymbol{AV_i}$**(x)** | Vom SC gebildeter Anfragevektor, welcher eine Anzahl von *SP* enthält, die vom SC zugegriffen werden, um $Q_i(x)$ zu bearbeiten. |
| $\boldsymbol{RV_i}$ | Von SC gebildeter Rückschreibevektor, welcher eingelesene Datensätze in geänderter Reihenfolge enthält. |
| **Shuffle(*AV*)** | Zufälliges Mischen der Einträge eines Vektors. |

| | |
|---|---|
| **Rand$^c$(*AV*)** | Zufälliges Auswählen von *c* Elementen ohne Wiederholung. |
| $\boldsymbol{MP_i}$ | Ein Pool im SC, welcher Card(*MP*) Datensätze zwischenspeichert; Inhalt zum Zeitpunkt *i*. |
| $\boldsymbol{H_i}$ | History-Liste, welche die Datensatznummern der letzten Card(*H*) Anfragen an den SC speichert; Inhalt zum Zeitpunkt *i*. |
| $\boldsymbol{c}$ | Sicherheitsfaktor, der die Größe der *AV* und somit die Anzahl der pro Anfrage zugegriffenen *SP* festlegt. |
| $\boldsymbol{c_h}$ | Anzahl der aus der History-Liste gewählten *SP* zur Bildung eines *AV*. |

Grundsätzlich arbeitet das Verfahren so, dass zur Bearbeitung einer Anfrage *c* Positionen der in Abschnitt 1 eingeführten gemischten und verschlüsselten Datenbank zugegriffen, der angefragte Datensatz *x* an den Nutzer ausgegeben und die gleichen *SP* in gleicher Reihenfolge noch einmal schreibend zugegriffen werden, um die eingelesenen *DS* in anderer Sortierung zurück in die Datenbank zu schreiben. Dabei wird zusätzlich ein Pool verwendet, so dass der angefragte Datensatz nicht unbedingt sofort wieder in die Datenbank geschrieben wird, sondern u.U. erst bei einer späteren Anfrage. Die Auswahl der zuzugreifenden *SP* folgt der Strategie, die in letzter Zeit zugegriffenen Datensätze gründlich zu durchmischen. Daher werden neben der Position des Datensatzes *x*, falls dieser sich nicht bereits im Pool befindet, $c_h$ Datensätze zufällig aus der History *H* und die restlichen Einträge des *AV* aus der gesamten DB ausgewählt:

**Schritt 1** *c* Speicherpositionen für den Zugriff auswählen (im SC)

$$x \notin H \wedge x \notin MP \;\; AV(x) = shuffle\left(SP(x), SP(rand^{c_h}(H)), SP(rand^{c-c_h-1}(DB))\right)$$
$$If \;\; x \in H \wedge x \notin MP \;\; AV(x) = shuffle\left(SP(x), SP(rand^{c_h-1}(H)), SP(rand^{c-c_h}(DB))\right)$$
$$x \in MP \qquad AV(x) = shuffle\left(SP(rand^{c_h}(H)), SP(rand^{c-c_h}(DB))\right)$$

**Schritt 2** Abruf der Speicherpositionen in *AV* aus der Datenbank durch das SC

Der SC ruft alle Positionen des Abfragevektors *AV*(*x*) in zufälliger Reihenfolge von der Datenbank ab, so das ein Beobachter nicht erkennen kann, an welcher der zugegriffenen Positionen *DS x* gespeichert war.

**Schritt 3** Ausgabe des angefragten Datensatzes an den Nutzer

**Schritt 4** Zurückschreiben der Datensätze in anderer Reihenfolge

Statt regelmäßig die gesamte Datenbank neu zu mischen, erfolgt eine partitielle Neumischung bei jeder Anfrage. Konkret werden die zugegriffenen Datensätze in neu gemischter und neu verschlüsselter Form[4] an die gleichen Datenbank-Positionen zurückgespeichert (siehe Abbildung 2), wobei zuvor der angefragte Datensatz mit einem DS aus dem Pool getauscht wird.

[4] Der neue Schlüsseltext darf an keiner Stelle mit dem bisherigen korrelieren. Daher ist ein indeterministisches Kryptoverfahren zu verwenden oder dem Klartext jeweils neu generierter Zufall hinzuzufügen.

$$SP = rand^{1}(MP_i, x)$$
$$MP_{i+1} = (MP_i \cup x) / SP$$
$$RV_i = shuffle(AV_i(x) / SP(x), SP)$$

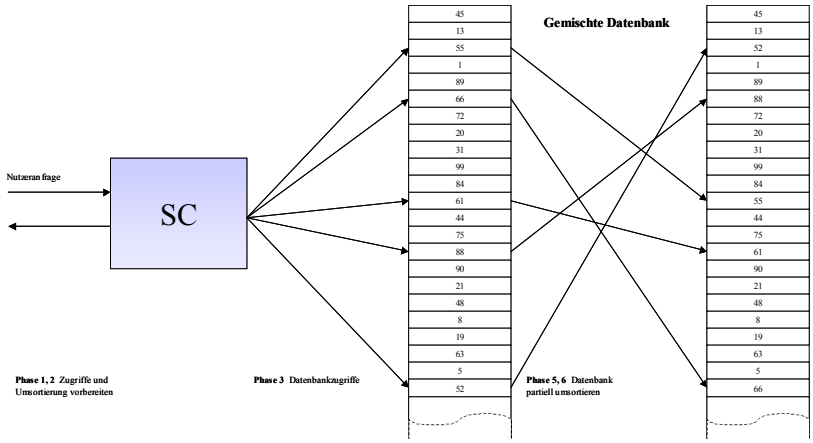


Abbildung 2: Bearbeitung einer Anfrage mit partiellen Durchmischen bei Poolgröße 1

Ebenso wie das Lesen eines Datensatzes lassen sich auch Schreiboperationen realisieren. Der Unterschied besteht darin, dass der jeweilige Datensatz nicht mit gleichem, sondern mit neuem Inhalt in den Pool geschrieben wird. Da die (spätere) Speicherung in der *DB* neu verschlüsselt erfolgt, kann der Angreifer diese Änderung nicht erkennen, daher auch Schreib- nicht von Lesezugriffen unterscheiden.

Welchen Vorteil bringt nun dieser Ansatz? Schon wenn kein Pool verwendet wird kann die Position des angefragten Datensatzes nur mit der Genauigkeit der Gruppe der *c* zugegriffenen *SP* ermittelt werden.

Bei Verwendung eines Pools kann der Angreifer keine sichere Aussage über die Speicherposition eines angefragten Datensatzes bei $Q_i$ treffen. In $AV_i$ befindet sich der Datensatz mit einer Wahrscheinlichkeit von $P$=1/Card($MP$). Mit exponentiell abnehmender Wahrscheinlichkeit kann sich der Datensatz $x$ hingegen auch in jedem der folgenden $AV_{j>i}$ befinden.

Ein rein passiver Beobachter erhält keine Information über den vom Nutzer tatsächlich abgefragten Datensatz, da die Datenbank verschlüsselt und durchmischt vorliegt und daher nur Zugriffe auf unbekannte Speicherpositionen beobachtet werden können. Durch passive Beobachtung könnten bestenfalls verschiedene Nutzeranfragen in Beziehung gesetzt werden, also Wahrscheinlichkeiten dafür ermittelt werden, das beide Anfragen denselben Datensatz betrafen.

Will der Angreifer Aussagen zu konkreten Datensätzen erhalten, muss er diese Datensätze vor oder nach einer beobachteten Nutzeranfrage selbst anfragen. Zum Erzielen von Unbeobachtbarkeit ist es daher wichtig, insbesondere bereits abgerufene Datensätze stark zu vermischen. Dies kann durch eine geschickte Bildung der *AV* geschehen, indem vermehrt Positionen zugegriffen werden, die vor kurzer Zeit angefragte Datensätze enthalten.

Dazu verwendet das Verfahren eine History-Liste *H*, welche die letzten Card(*H*) Anfragen protokolliert. Die Parameter Card(*H*), Card(*MP*) und $c_h$ sollten so gewählt werden, dass eine bei einer Nutzeranfrage $Q_{j(>i)}$ beobachtete Überschneidung von $AV_j(?)$ mit $AV_i(x)$, den Zugriffspositionen einer früheren Anfrage des Angreifers nach Datensatz *x*, keine signifikante Wahrscheinlichkeits-aussage dafür ergibt, dass bei der Nutzeranfrage $Q_j$ Datensatz *x* zugegriffen wurde. Um dies sicherzustellen sollte die Wahrscheinlichkeit, dass der Zugriff auf $AV_i(x)$ zufällig durch Wahl der $c_x$ Positionen aus der History geschah, zumindest genauso groß sein, wie die Wahrscheinlichkeit, dass sich Datensatz *x* an einer Position von $AV_i(x)$ befindet.

Die linke Seite der Formel repräsentiert die zufällige Auswahl einer Position aus $AV_i$. Zur Bearbeitung von $Q_j$ werden dafür $c_x$ Elemente aus der History ausgewählt. Um eine Überschneidung zu erzielen, muss mindestens einmal einer der $c_x$ Einträge ausgewählt werden, die schon in Anfrage $Q_i$ ausgewählt wurden.

$$\frac{c_x^{\,2}}{Card(H)} \geq \frac{1}{Card(MP)}$$

Die Beobachtung der Speicherzugriffe ermöglicht somit keine signifikante[5] Wahrscheinlichkeitsaussage über die Anfrage bezüglich der *DS* der letzten Card(*H*) Anfragen. Zudem wird jeder Eintrag in *H* im Mittel $c_x$ Mal erneut zugegriffen und dabei seine Position jeweils mit $c_x$-1 anderen Datensätzen aus *H* vermischt, so dass die Beobachtung der Zugriffe einer viel späteren Anfrage $Q_j$ auf ein $AV_{i<<j\text{-Card}(H)}$ neben einer erheblichen Anfragewahrscheinlichkeit für jeden beliebigen Datensatz der Datenbank Card(*H*) nahezu gleichwahrscheinliche Datensätze als mögliche Anfrage $Q_j$ ermittelt.

Im folgenden Abschnitt wird gezeigt, dass dieses Basisverfahren mit einem $c \geq 15$, Card(*MP*)=10 und Card(*H*)=1000 unabhängig von der Datenbankgröße ausreichende Sicherheit bietet. Damit sind pro Anfrage nur 30 Lese- bzw. Schreibzugriffe auf den Speicher erforderlich ($O(1)$). Die im Abschnitt 2 beschriebenen perfekten Verfahren benötigen inkl. Neumischen bei [AsFr02] durchschnittlich $O(N)$ Zugriffe pro Anfrage.

---

[5] Im Falle der Anfrage nach DS *x* in $Q_j$ addieren sich die Wahrscheinlichkeiten für den zufälligen und den gezielten Zugriff auf $AV_i(x)$. Um identische Wahrscheinlichkeiten zu erzielen kann in der History zusätzlich die jeweiligen *SP* der *AV* speichern und die Zufallsauswahl bei wiederholter Anfrage nach DS *x* verhindern.

## 3.2 Unbeobachtbarkeitsmaß

Absolute Unbeobachtbarkeit im Kontext von PDA-Datenbanken bedeutet, dass bei jeder Anfrage und nach jeder dem Angreifer möglichen Beobachtung jeder Datensatz der gesamten Datenbank mit der gleichen Wahrscheinlichkeit $P$=1/$N$ als der angefragte in Frage kommt.

Als Bewertungsmaß für die erzielte Unbeobachtbarkeit wird in [DSCP02] die Shannonsche Entropie vorgeschlagen, wobei $P_i$ die Anfragewahrscheinlichkeit eines bestimmten Datensatzes ist.

$$E = -\sum_{i=1}^{N} P_i * ld\ P_i \quad Perfektes\ System: \quad E = ld\ N$$

Bei perfekten Verfahren ist die Unbeobachtbarkeit ausschließlich von der Anzahl der Möglichkeiten (DS der Datenbank) abhängig.

Bei probabilistischen Verfahren werden unterschiedliche Anfragewahrscheinlichkeiten toleriert. Es wird jedoch oft gefordert, dass kein Datensatz aus der potentiellen Abfragemenge ausgeschlossen werden kann. Die Anzahl der Möglichkeiten ist daher genauso groß wie bei perfekten Verfahren, aber die Unsicherheit des Beobachters, die Entropie, auf Grund geringerer Wahrscheinlichkeiten für einen Teil der Möglichkeiten geringer.

Ein Verfahren dieser Art [Ason03] wurde im letzten Abschnitt vorgestellt. Dabei erfolgt die Bearbeitung einer Anfrage mit wahlfreien Zugriff auf $a$ Datensätze der Originaldatenbank und b sequentiellen Zugriffen auf eine gemischte Datenbank. Bei $b<<N$ ist daher die Wahrscheinlichkeit sehr groß, dass der gesuchte Datensatz von der Originaldatenbank angefragt wurde. Es müssen folglich zwei Gruppen von Auftrittswahrscheinlichkeiten unterschieden werden: Die bei $a$ zugegriffenen Datensätze und alle restlichen Datensätze der Datenbank. Folgendermaßen berechnet sich die Entropie:

$$E = \left(1 - \frac{b}{N}\right) * ld\left(\frac{a}{1 - b/N}\right) + \frac{b}{N} * ld\left(\frac{N * (N - a)}{b}\right)$$

Im folgenden wird zum Vergleich die Entropie des in diesem Artikel vorgestellten Verfahrens berechnet. Wie schon beschrieben kann nur durch aktive Angriffe, konkret das vorherige oder nachfolgende Abrufen der zu beobachtenden Datensätze durch den Angreifer überhaupt eine Aussage bezüglich des in einer Nutzeranfrage angefragten *DS* ermittelt werden. Durch geschickte Wahl der Systemparameter wird erreicht, dass über die letzten Card($H$) angefragten Datensätze keine signifikante Aussage getroffen werden kann. Jeder dieser Datensätze, aber auch jeder andere ist ein mögliches Ziel der beobachteten Anfrage, da pro Anfrage $c_x$ DS aus $H$ aber auch $c$-$c_x$ andere Datensätze zugegriffen werden.

Zur Untersuchung der $c$-$c_x$ Zugriffe muss man sich verdeutlichen, dass jeder dieser Zugriffe zu einem Bereich der Datenbank führen könnte, über den der Angreifer Aussagen treffen kann. Dies ist dann der Fall, wenn die zugegriffene Speicherposition bereits bei einer früheren Anfrage zugegriffen wurde, die der Angreifer beobachtet oder besser selbst ausgeführt hat.

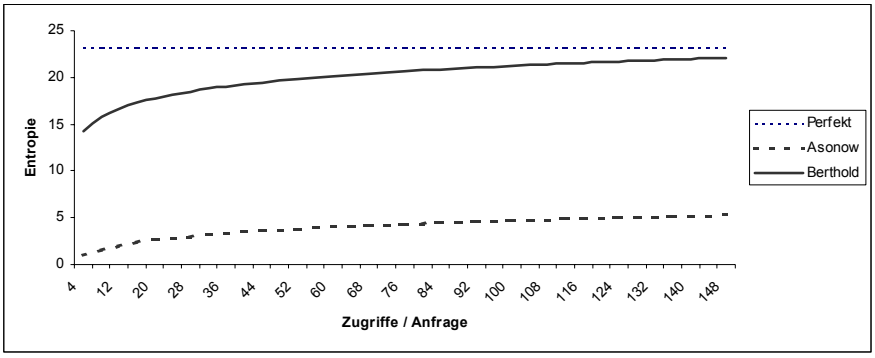


Abbildung 3: Vergleich des Unbeobachtbarkeitsmaßes verschiedener PDA-Verfahren, bei $N$=10 Mio *DS,* Card($MP$)=10. Bei den Verfahren von Asonow wurde der zusätzliche Aufwand für das regelmäßige Neumischen nicht berücksichtigt.

Zur Berechnung der Entropie soll vom schlimmsten Fall ausgegangen werden. Der Angreifer kann jede der $c$-$c_x$ Positionen früheren Anfragen zuordnen, welche er selbst ausgeführt hat. Zudem hat der Angreifer die jeweils Card($H$) Anfragen vor den $c$-$c_x$ früheren Anfragen selbst ausgeführt, so dass der Inhalt der History, also die sich möglicherweise in den betrachteten *AV*'s befindenden Datensätze dem Angreifer bekannt sind.

Da man mit hoher Wahrscheinlichkeit annehmen kann, dass die in den jeweiligenen Anfragen angefragten Datensätze auf Grund der History später wiederholt zugegriffen und deren Positionen daher verändert wurden, werden diese hier nicht gesondert behandelt. Daher ist der Angreifer in der Lage, die ($c$-$c_x$)*Card($H$) als wahrscheinliche Ziele der Anfrage zu identifizieren. Die Wahrscheinlichkeit, dass sich an den zugegriffenen Positionen einer dieser Datensätze aus dem jeweiligen $H$ befindet beträgt P=$c_x/c$, da auch damals $c$-$c_x$ *DS* von beliebigen Positionen der Datenbank zugegriffen wurden. Folglich stehen für die Entropieberechnung zwei Gruppen von Datensätzen mit unterschiedlichen Wahrscheinlichkeiten gegenüber: ($c$-$c_x$)*Card($H$) DS mit $P_{gesamt}= c_x/c$ und alle restlichen Datensätze mit $P_{gesamt}= (c-c_x)/c$.

$$E = \frac{c_x}{c} * ld\left(\frac{c(c-c_x) * Card(H)}{c_x}\right) + \frac{c-c_x}{c} * ld\left(\frac{c(N-(c-c_x) * Card(H))}{c-c_x}\right)$$

## 3.3 Angriffsaufwand

Neben dem reinen Vergleich des maximal möglichen Beobachtungserfolges soll hier der notwendige Aufwand für die Durchführung einer Überwachung dargestellt werden. Bei dem Verfahren von Asonov genügt es, die Zugriffe auf die Original-Datenbank zu beobachten und die Parameter a und b zu kennen, um jeden der $a$ von der Originaldatenbank zugegriffenen Datensätze mit der Wahrscheinlichkeit $P = (1 - b/N)/a$ als das Ziel der Anfrage zu identifizieren.

Für das in diesem Artikel beschriebene Verfahren muss der Angreifer jeden einzelnen Datensatz, über den er eine Aussage wünscht, selbst mindestens einmal anfragen. Zum Erreichen des obigen Beobachtungserfolges wäre die Anfrage eines erheblichen Teils der Datenbank erforderlich.

Der Versuch einige wenige Datensätze zu beobachten erfordert hingegen nur den Aufwand, eben diese Datensätze in größeren Abständen einmal anzufragen. Allerdings ist der Beobachtungserfolg gering: Beobachtet der Angreifer einen Zugriff auf den damaligen $AV$, bzw. auf eine der vielen Positionen, wohin Positionen des damaligen $AV$ getauscht worden sein könnten, ist die bestmögliche[6] Wahrscheinlichkeitsaussage $P = c_x/c * 1/Card(H)$ dafür, dass der gesuchte Datensatz angefragt wurde.

## 3.4 Ausschluss von Datensätzen

Bisher nicht adressiert wurde die Problematik, dass der Angreifer ggf. Datensätze aus der potentiellen Anfragemenge ausschließen kann. Dies zu vermeiden ist jedoch eine wesentliche Sicherheitseigenschaft eines PDA-Verfahrens. Durch die Verwendung eines Pools ist es nicht möglich, bisher angefragte Datensätze auszuschließen, da ein Datensatz potentiell unendlich lang im Pool verbleiben kann. Zudem erfolgt durch die History ein häufiger Zugriff auf bisher angefragte Datensätze. Jedoch kann man sich Folgen von Anfragen und Datenbankzugriffen geben, die die Anzahl der möglichen Datensätze auf einen Bruchteil der Datenbank einschränkt und so natürlich die Unbeobachtbarkeit ad absurdum führt.

Wenn bei $c$ verschiedenen Anfragen des Angreifers beispielsweise zufällig immer das gleiche $AV$ abgefragt würde, kennt der Angreifer alle an den Positionen des $AV$ bzw. im Pool gespeicherten Datensätze. Erfolgt nun wiederum eine Nutzeranfrage mit dem gleichen $AV$, wüsste der Angreifer, dass nur die $c$ bisher abgefragten Datensätze statt aller Datensätze der Datenbank als Anfrage möglich waren.

---

6 Die Wahrscheinlichkeit ist nur dann so hoch, wenn alle $c$-$c_x$ Zugriffe der beobachteten Anfrage zufällig auf Positionen erfolgen, die in früheren kurz aufeinanderfolgenden Anfragen zugegriffen wurden und daher die gleichen Historyeinträge verwenden.

Dies lässt sich sehr einfach verhindern: Derartige Angriffe sind nur möglich, wenn $i$+1 Anfragen innerhalb einer Teilmenge der $T \subset DB$ mit $i$=Card($T$) bleiben, der Nutzer im $i$+1'ten Zugriff also einen der bisher vom Angreifer abgerufenen Datensätze nochmals abruft. Verhindert werden kann dies, indem eine Position des jeweiligen $AV$ sequentiell durch die Datenbank geht. Dadurch überstreichen $i$ Anfragen auch mindestens eine Menge von $i$ Positionen, bis Card($T$)=$N$ ist.

Durch obigen Angriff hat der Angreifer alle Datensätze ermittelt, die in einem bestimmten Teil der Datenbank gespeichert sind. Weitere Möglichkeiten zum Ausschluss von Datensätzen gibt es nicht: Jede entsprechend des Angriffs nicht vollständig determinierte Position könnte jeden bisher nicht vom Angreifer angefragten Datensatz enthalten. Zugleich kann die Anfrage wegen des Pools jeden bisher angefragten Datensatz betreffen.

**Das PDA-Verfahren verhindert den Ausschluss von Datensätzen aus der potentiellen Anfragemenge. Daher kann jeder Nutzerzugriff potentiell jeden Datensatz der gesamten Datenbank zum Ziel haben.**

# 4 Diskussion

Es sollte ein Verfahren vorgestellt werden, welches möglichst starke Unbeobachbarkeit der Zugriffe auf ein PDA-System ermöglicht. Das ausreichende Unbeobachtbarkeit erzielt werden kann, wurde bereits gezeigt. Zu diskutieren ist noch, ob das Verfahren praktisch einsetzbar ist, also ob der Aufwand nicht zu hoch ist, die Beschränkungen der SC den Anforderungen genügen und auch Datensicherheit gewährleistet werden kann.

## 4.1 Aufbau der Datenbank

Bisher wurde angegeben, dass das unbeobachtbare Mischen einer Datenbank mit Hilfe des SC einen Aufwand $O(N*\log N)$ verursacht. Verwendet man das gleiche probabilistische Verfahren wie zum Abfragen der Datenbank auch zum Aufbau derselben, beträgt der Aufwand $O(N)$. Das Hinzufügen eines Datensatzes erfolgt dabei ähnlich einer Anfrage: Es werden $c$ Datensätze gelesen und $c$+1 Datensätze geschrieben, wobei $c$ Datensätze an die bisherigen Positionen geschrieben werden und ein Datensatz an die Datenbank angehängt wird. Der neue Datensatz kann sich wegen der Durchmischung nachher an einer der $c$+1 Positionen oder noch im Pool befinden. Das Hinzufügen eines Datensatzes ist dadurch im laufenden Betrieb mit $O(1)$ problemlos möglich.

## 4.2 Aufwand

Der Aufwand für die Bearbeitung einer Anfrage beträgt $c$ Lese- und $c$ Schreibzugriffe des SC auf die Datenbank. Dabei wird vorausgesetzt, dass der SC mindestens ($c$-1)+Card($MP$) Datensätze zwischenspeichern kann. Es ist sinnvoll, die Datensatzlänge $l$ so zu wählen, dass dies ermöglicht wird. Andernfalls entsteht für das unbeobachtbare Umsortieren ein Aufwand von $O(c*MP*\log c)$ [Ason03]. Der Aufwand ist somit unabhängig von der Datenbankgröße. Die Berechnungen haben ergeben, dass $c \in [15,25]$ ausreichende Sicherheit bietet. Der SC benötigt internen Speicher zum Ablegen der Umsortierungstabelle. Dazu ist pro Datensatz die Datenbankposition zu speichern ($O(N)$) und zusätzlich müssen relativ wenige O(Card($H$)) Einträge der History gespeichert werden.

Dieser Speicheraufwand ist problematisch, jedoch beträgt der notwendige Speicherplatz nur ld($N$)*$N$ Bit, also nur einen Bruchteil des Speicherplatzes der Datenbank ($N*l$). Zudem bestehen zwei Möglichkeiten: Entweder wird die Tabelle auf mehrere SC's verteilt, die bei großen Datenbanken schon zur Bewältigung der Anfragelast notwendig sind, oder extern verschlüsselt abgelegt und bei jedem Zugriff eingelesen, so dass der Speicheraufwand zu Lasten des Berechnungsaufwandes reduziert werden kann.

Zudem muss das SC pro Anfrage $c*l$ Bytes mit einer symmetrischen Chiffre entschlüsseln und mit einem neuen Initialwert wieder verschlüsseln.

## 4.3 Parallelisierung

In Abschnitt 3 wird beschrieben, wie das probabilistische PDA-Verfahren mit einer Datenbank und einem SC realisiert werden kann. Für den praktischen Einsatz insbesondere bei vielen Nutzern ist dies nicht ausreichend. Daher werden hier einige Möglichkeiten der Parallelisierung erörtert.

Eine Möglichkeit ist die mehrfache Duplizierung des gesamten Systems. Soll die Datenbank ausschließlich Informationen anbieten, d.h. nicht durch die Nutzer geändert werden, ist dies problemlos möglich. Insbesondere wird der Grad der Unbeobachtbarkeit nicht eingeschränkt, da PDA-Datenbanken unabhängig von einer Anonymitätsgruppe arbeiten. Sind Datenveränderungen durch Nutzerzugriffe vorgesehen, so müssen diese synchron auf allen Datenbanken erfolgen. Eine Duplizierung lohnt sich daher nur, wenn Datenänderungen vergleichsweise selten auftreten.

Eine weitere Möglichkeit ist die Duplizierung des SC, wobei die gleiche Datenbank genutzt wird[7].

[7] Als entscheidender Flaschenhals wurde in [SS00] die Geschwindigkeit der SC bzw. deren Ein- und Ausgabeschnittstelle identifiziert. Die Geschwindigkeit des

Da sich die Tabelle, welche die Position der Datensätze speichert, schlecht verteilt auf mehrere SC verwalten lässt, schlagen wir vor, einen $SC_1$ zur Verwaltung dieser Tabelle einzusetzen und alle anderen $SC_{2\text{-}m}$ zur Ausführung der Nutzeranfragen[8].

Zudem ist ein verteiltes Poolen von Datensätzen in den einzelnen SC problematisch. Neben höherem Synchronisationsaufwand wäre auf diese Weise mit höherer Wahrscheinlichkeit erkennbar, wann ein Datensatz einen Pool verlässt. Eine Alternative wäre die zentrale Speicherung des Pools, welche jedoch den Umfang der möglichen Parallelisierung begrenzt, da pro Anfrage der Pool-SC jeweils einen Datensatz empfangen und ausgeben muss.

## 4.4 Datensicherheit

Für die langfristige Sicherheit der Daten ist es ein Problem, dass allein der SC über den notwendigen Schlüssel zum Entschlüsseln der Datenbank verfügt. Dieses Problem kann durch die Nutzung mehrerer SC entschärft werden. Für die Parallelisierung ist es in jedem Fall notwendig, den Schlüssel zwischen den SC's zu übertragen. Dies kann sicher geschehen, indem ein SC den Schlüssel nur dann weitergibt, wenn das empfangende SC nachweist, dass das gleiche Programm wie im sendenden SC läuft. Es könnte ein Backup-SC den Schlüssel erhalten und an einem sicheren Ort verwahrt werden.

Ein weiteres Problem ist die Frage der Backup-Fähigkeit, wenn per Nutzerzugriff eine Änderung der Datenbank erfolgen kann. Ein regelmäßiges Gesamtbackup der Datenbank erscheint nicht praktikabel. Jedoch könnte ein Differenzbackup durchgeführt werden. Die SC könnten regelmäßig geänderte Datensätze in einen separaten Speicher sichern und bei einem Ausfall mit diesen Datensätzen das letzte Gesamtbackup aktualisieren. Für besonders kritische Dienste, wie beispielsweise den Paymentservice, könnte ein sofortiges Backup vorgesehen werden. Bei jeder Anfrage würde dann ein zusätzlicher Datensatz in das Backupmedium geschrieben[9]. So kann eine verlustfreie Wiederherstellung der Datenbank sichergestellt werden.

---

Sekundärspeichers lässt sich durch bekannte Mechanismen (z.B. RAID-System) problemlos erhöhen.

[8] Der Ablauf einer Anfrage ist beispielsweise folgender: Die Nutzeranfrage wird über einen Lastverteilungsmechanismus an einen der $SC_{2\text{-}m}$ weitergeleitet. Der SC ermittelt den abzufragenden *DS* und schickt diese Information an $SC_1$. $SC_1$ ermittelt den Anfragevektor *AV*, gibt die Umsortierung vor und sperrt die betreffenden Datensätze, bis der SC die Ausführung der Anfrage meldet.

[9] Um die Unterscheidbarkeit von Schreib- und Lesezugriffen zu vermeiden müsste dies bei beiden Zugriffsarten geschehen.

# 5 Zusammenfassung

In diesem Artikel wurde ein neues Verfahren zur effizienten Lösung des Problems des privaten Datenbankzugriffs vorgestellt. Der Ansatz sieht einerseits die Nutzung sicherer Hardware und andererseits eine Beschränkung auf probabilistische Sicherheit vor. Dadurch sind trotz eines hohen Grades an Unbeobachtbarheit Datenbankzugriffe mit einem Berechnungsaufwand von $O(1)$ möglich. Auch der Kommunikationsaufwand ist optimal, da einzig der angefragte Datensatz zum Nutzer gesendet wird und die Größe der Anfrage zu vernachlässigen ist. Im Vergleich zu anderen Verfahren sind flexiblere Dienste realisierbar, insbesondere weil das vorgestellte Verfahren als bisher einziges ein unbeobachtbares Ändern der Datenbank erlaubt. Die Beschränkung der Sicherheit hat die Auswirkung, dass Vertrauen in die Sicherheit eines physisch geschützten Gerätes gelegt werden muss. Zudem kann ein mächtiger Beobachter u.U. verschiedene Anfragewahrscheinlichkeiten für Teile der Datenbank ermitteln. Es kann jedoch sichergestellt werden, dass der Angreifer keinen Datensatz bei einer Anfrage ausschließen kann. Zusätzlich ist der „richtige“ Verdacht des Angreifers Voraussetzung für einen Überwachungserfolg, da nur ein Teil der Datenbank überwacht werden kann. Bei perfekten PIR-Verfahren kann der Angreifer keinerlei Wahrscheinlichkeitsunterschiede ermitteln, dafür muss man aber einen Berechnungsaufwand von $O(N)$ in Kauf nehmen.

Noch zu untersuchen sind die Auswirkungen des Abrufes verschiedener Daten, welche unterschiedlich viele Zugriffe benötigen. In [KAPe03] werden derartige Auswirkungen untersucht und gezeigt, dass schon anhand der Datenlänge Rückschlüsse auf die jeweils abgerufene Information möglich sein können. In diesem Zusammenhang müssen die Auswirkungen der Wahrscheinlichkeitsunterschiede untersucht werden.

# Literatur

[CGKS95] Benny Chor, Oded Goldreich, Eyal Kushilevitz, Madhu Sudan: Private information retrieval; Proceedings of 36th FOCS, 1995.

[CoBi_95] David A. Cooper, Kenneth P. Birman: Preserving Privacy in a Network of Mobile Computers; 1995 IEEE Symposium on Research in Security and Privacy, IEEE Computer Society Press, Los Alamitos 1995, 26-38.

[KO97] E. Kushilevitz, R. Ostrovsky: Replication is not needed: single database, computationally-private information retrieval; FOCS'97, pp. 364-373.

[CGN97] Benny Chor, Niv Gilboa, Moni Naor: Private information retrieval by keywords; Technical report, Technion - Israel Institute of Technology, 1997.

[Gil00] Niv Gilboa: Topics in Private Information Retrieval; PhD thesis, Technion - Israel Institute of Technology, 2000.

[SS00] Sean W. Smith, Dave Safford: Practical server privacy with secure coprocessors. IBM Systems Journal, 40(3), September 2001.

[PIP00] Susannah Fox: Trust and privacy online: Why Americans want to rewrite the rules. The Internet Life Report, The Pew Internet & American Life Project, 2000, http://www.pewinternet.org/pdfs/PIP_Trust_Privacy_Report.pdf.

[BFKö01] Oliver Berthold, Hannes Federrath, Stefan Köpsell: Web MIXes: A system for anonymous and unobservable Internet access. in: Hannes Federrath (Ed.): Designing Privacy Enhancing Technologies. Proc. Workshop on Design Issues in Anonymity and Unobservability, LNCS 2009, Springer-Verlag, Heidelberg 2001, 115-129.

[BCKP01] Oliver Berthold, Sebastian Clauß, Stefan Köpsell, Andreas Pfitzmann: Efficiency Improvements of the Private Message Service. in: Information Hiding 4th InternationalWorkshop, IH 2001 Pittsburg, PA, USA, April 2001 Proceedings, LNCS 2137, Springer-Verlag Berlin, 112-125.

[AsFr02] Dmitri Asonov; Johann-Christoph Freytag: Almost optimal private information retrieval. In: Proceedings of 2nd Workshop on Privacy Enhancing Technologies (PET 2003), San Francisco, USA, April 2002.

[HILM02] Hakan Hacigümüs, Balakrishna R. Iyer, Chen Li, Sharad Mehrotra: Executing SQL over Encrypted Data in the Database-Service-Provider Model. SIGMOD Conference 2002: 216-227.

[DSCP02] Claudia Díaz, Stefaan Seys, Joris Claessens and Bart Preneel: Towards measuring anonymity. Proceedings of PET 2002, San Francisco (USA). Dingledine and Syverson (Eds.), Designing Privacy Enhancing Technologies, LNCS 2482, pp. 54-68

[Ason03] Dmitri Asonov: Querying Databases Privately. Dissertation, Humboldt University Berlin, March 2003.

[KAPe03] Dogan Kesdogan, Dakshi Agrawal, Stefan Penz: Limits of Anonymity in Open Environments. Information Hiding: 5th International Workshop, Springer-Verlag, 2003

[Chan04] Yan-Cheng Chang: Single Database Private Information Retrieval with Logarithmic Communication; Cryptology ePrint Archive, Report 2004/036, 2004, http://eprint.iacr.org.