Köhler *et al. Journal of Cloud Computing: Advances, Systems and Applications* (2015) 4:1
DOI 10.1186/s13677-014-0025-1

Journal of Cloud Computing
a SpringerOpen Journal

RESEARCH Open Access

Confidential database-as-a-service approaches: taxonomy and survey

Jens Köhler*, Konrad Jünemann and Hannes Hartenstein

Abstract

Outsourcing data to external providers has gained momentum with the advent of cloud computing. Encryption allows data confidentiality to be preserved when outsourcing data to untrusted external providers that may be compromised by attackers. However, encryption has to be applied in a way that still allows the external provider to evaluate queries received from the client. Even though confidential database-as-a-service (DaaS) is still an active field of research, various techniques already address this problem, which we call *confidentiality preserving indexing approaches* (CPIs). CPIs make individual tradeoffs between the functionality provided, i.e., the types of queries that can be evaluated, the level of protection achieved, and performance.

In this paper, we present a taxonomy of requirements that CPIs have to satisfy in deployment scenarios including the required functionality and the required level of protection against various attackers. We show that the taxonomy's underlying principles serve as a methodology to assess CPIs, primarily by linking attacker models to CPI security properties. By use of this methodology, we survey and assess ten previously proposed CPIs. The resulting CPI catalog can help the reader who would like to build DaaS solutions to facilitate DaaS design decisions while the proposed taxonomy and methodology can also be applied to assess upcoming CPI approaches.

Keywords: Database-as-a-service, Confidential data outsourcing, Confidentiality preserving indexes, Attacker models, Survey, Taxonomy

Introduction

The cloud computing paradigm promises benefits such as cost-effectiveness and scalability with regard to outsourcing services and data to external providers. However, loss of control over the outsourced data puts data confidentiality at risk – in particular in the database-as-a-service scenario (DaaS), where databases are outsourced to external *storage providers* (SPs), who could potentially be interested in the content of the outsourced data. Other attack scenarios include curious administrators who work for the SP or external attackers.

One way to address this problem is to thoroughly assess whether an SP can be trusted to enforce data confidentiality [1]. However, making such assessments is hard and in many cases users do not consider any SPs trustworthy [2]. The approach that we investigate further in this paper assumes that all SPs may be compromised. Thus data confidentiality is enforced by encrypting the data before it is outsourced.

To efficiently retrieve data records that match a certain query, traditional database systems make use of indexes. While the data records can be encrypted to preserve confidentiality, the records in traditional indexes have to be in plaintext in order to allow efficient query execution. In order to protect data confidentiality in the indexes, several authors propose *confidentiality preserving indexing approaches* (CPIs), such as the deterministic encryption or hashing of data values to be outsourced. The setting in which these approaches operate is shown in Figure 1. In this scenario, a user wishes to outsource data to one or more SPs. While the trusted user may view the data, the external SPs are considered untrustworthy or prone to being compromised by third-party attackers. Therefore, the data has to be protected before being outsourced, in order to preserve data confidentiality. In order to protect the data, the user runs a trusted *mediator* that encrypts

*Correspondence: jens.koehler@kit.edu
Steinbuch Centre for Computing (SCC), Karlsruhe Institute of Technology (KIT), Zirkel 2, Building 20.21, 76131 Karlsruhe, Germany

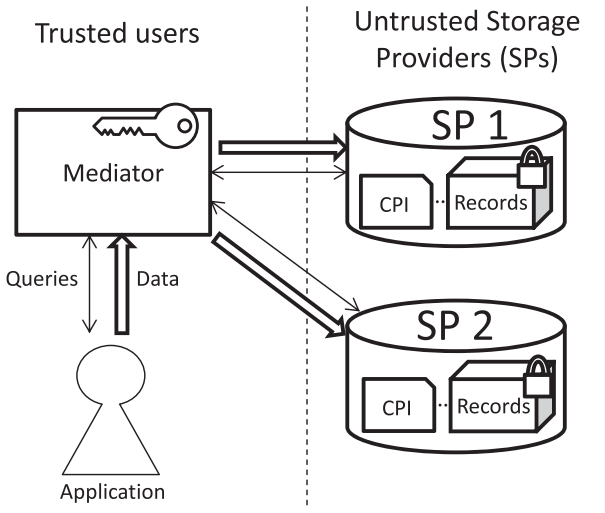

Figure 1 The database-as-a-service setting.

the data records before outsourcing them and applies CPI approaches to build confidentiality-preserving indexes that can be used to evaluate queries efficiently.

Problem statement: *Choosing CPIs that are suited to confidentially outsource data in a given deployment scenario represents a challenge: Each CPI supports specific query types and provides specific security properties. In a deployment scenario, requirements exist with regard to the queries that have to be executable, the attackers to protect against and the required protection level, i.e., whether or not it is sufficient to protect only a subset of all the outsourced data items. In particular, assessing which attackers a CPI protects against is not trivial because most publications list security properties of their approaches rather than the addressed attacker models in a way that can be compared to other approaches.*

In this paper, we present a taxonomy of requirements that CPIs have to satisfy in various deployment scenarios including the required functionality and the level of protection against various attackers. Based on the taxonomy, we propose a methodology to assess the applicability of CPIs in deployment scenarios by linking attacker models to CPI security properties. We apply this methodology to a survey covering ten of the most popular CPIs and build a *CPI catalog* that makes it easy to determine whether a specific CPI suits the security requirements of a particular deployment scenario. Compared to other surveys such as [3], we focus less on identifying future research challenges in the DaaS domain and more on the applicability of existing CPIs in different deployment scenarios. Therefore, we refer to the original publications and other surveys for an in-depth illustration of the cryptographic foundation and the background of individual CPIs.

The focus of this study is on preserving the confidentiality of outsourced relational data, i.e., data that contains *records* consisting of a fixed set of *attribute values*. In particular, we do not address related areas such as the preservation of data integrity [4], proof of retrievability [5] or access control enforcement [6] in a multiuser setting. These problem settings are orthogonal to preserving confidentiality and can be applied in addition to the approaches we investigate in this paper.

The main contributions of this paper are:

- A **taxonomy** of deployment scenario requirements with regard to the required functionality, assumed attackers and protection levels that have to be guaranteed when outsourcing data.
- A **methodology** for assessing which taxonomy requirements are satisfied by a given CPI approach. To assess a CPI, the attacker capabilities the approach protects against have to be derived from a set of *security properties* the CPI satisfies. A mapping between the security properties and attacker capabilities is provided in this paper.
- A **survey** of popular CPIs that applies the proposed methodology to build a **CPI catalog** based on the proposed taxonomy. The CPI catalog makes it possible to quickly determine which CPIs can satisfy the requirements of a specific deployment scenario. Furthermore, we provide a high-level **performance evaluation** of the presented CPIs.

The paper is structured as follows: First, we present the taxonomy of deployment scenario requirements on CPI approaches. Then we provide the methodology to assess which requirements a CPI satisfies with regard to the taxonomy and use the proposed methodology to categorize the surveyed CPI approaches. Finally, we draw the key conclusions and highlight future research directions.

Taxonomy of deployment requirements

A CPI approach that is used in a specific deployment scenario has to satisfy multiple scenario-specific requirements with regard to (1) the supported *functionality*, i.e., which queries can be efficiently executed and how outsourced data can be modified, (2) the *protection level*, i.e., a definition of when the data can be considered protected and (3) the ability to protect the data confidentiality against the assumed *attackers* and their capabilities. From a user's perspective, determining the functionality requirements is usually easier than determining the protection goal and the assumed attacker capabilities. However, our study underlines the importance of paying attention to all three dimensions. The proposed taxonomy constitutes a foundation that supports developers of frameworks that use CPIs in creating

a concise description of their individual deployment requirements [7,8].

In the following, we introduce a taxonomy of the deployment scenario requirements. We list the functionality requirements in addition to categorizing protection levels and attacker models.

Functionality

In most deployment scenarios, it is important for a CPI approach to support efficient query execution on the outsourced data. Queries can be used to retrieve parts of the outsourced data that match certain criteria. For the sake of simplicity and readability, we focus on a subset of Structured Query Language (SQL). However, the study can be extended with additional functionalities based on the proposed methodology by checking for each CPI whether it supports the functionality. For instance, Join operations are supported by deterministic indexes and order-preserving encryption. We distinguish between three ways to filter records:

Equality selections (ES) can be used to retrieve records that have a certain value for an attribute. *Example:* The query `SELECT...WHERE name='Adam'` performs an equality selection on the attribute *name*.

Range selections (RS) can be used to retrieve records that have an attribute value that lies within a certain range. *Example:* The query `SELECT...WHERE age<30` performs a range selection on the attribute *age*.

Like selections (LS) can be used to retrieve records that have an attribute value that is similar to a certain search term. *Example:* The query `SELECT...WHERE name LIKE 'Ad'` performs a like selection on the attribute *name*.

Furthermore, **aggregation (AG)** queries do not retrieve records but request aggregated attribute values of multiple records. *Example:* The query `SELECT SUM(salary)...` performs an aggregation on the attribute *salary*.

Besides evaluating queries efficiently, many deployment scenarios also require the modification of outsourced data. While some proposed CPI approaches assume that the data is outsourced once without being modified at a future point in time, other approaches allow changes to be made to the outsourced data without harming its confidentiality. Modifications of the data may constitute **insertions**, **updates** or **deletions** of records.

Typically, a CPI supports specific queries. Thus, future queries have to be known before the data is outsourced. Outsourcing frameworks [8] that use CPIs can support ad-hoc queries by using onion encryption and apply all available CPIs by default when outsourcing data. However, as support for ad-hoc queries is not a feature of CPIs, we consider it out of this article's scope.

Protection levels

In this paper, we focus on preserving the confidentiality of records in the outsourced data. A confidentiality requirement on a record level can be that specific attributes of a record must not be readable by an attacker or that *specific* attributes must not be readable *together* by an attacker. Consider, for instance, the data shown in Table 1. By (partially) encrypting the contained values as shown in Table 2 one can prevent an attacker from determining the salary of record α, as the attribute *salary* contains indistinguishable ciphertexts only. We do *not* aim to preserve the confidentiality of attributes in general, i.e., CPIs may still leak information on single attributes such as their frequency distribution. For instance, in Table 2, the *gender* attribute is encrypted but distinguishable. Thus, an attacker does not learn the *gender* plaintext values of the records but the frequency distribution of attributes.

Example: In Table 1, a confidentiality requirement can be that an attacker must be unable to map a *salary* to a *name*. For instance, this requirement can be satisfied by protecting the *salary* attribute by encrypting it or by not storing it in indexes at all. Thus, an attacker is able to read the name of an outsourced record but not the salary.

The example shows that attribute combinations can be protected by protecting specific attributes. In the following we denote such attributes as *protected attributes*.

Existing approaches can be categorized according to their notion of *record protection*, i.e., when they consider the records to be sufficiently protected:

- **Computational record protection**: The outsourced records are considered protected if computationally bounded attackers with access to the outsourced database cannot discover anything about the confidential information contained in each record that they would not know without having access to the database.
- **Probabilistic record protection**: A CPI ensures only probabilistic record protection if an attacker might be able to infer confidential attribute combinations of a small set of records or narrow down the possible confidential attribute combinations of certain records. For the sake of

Table 1 Running example – plaintext data

ID	Name	Rating	Gender	Salary	Age
1	Adam	1	m	20000	23
2	Bob	2	m	20000	25
3	Carol	3	f	24000	25
4	Dan	4	m	20000	30
5	Eve	5	f	26000	30
6	Adam	6	m	40000	30

Table 2 Running example – outsourced data

ID	Name	Rating	Gender	Salary	ID	Age
α	Adam	345	η	ι	π	23
β	Bob	452	η	κ	ρ	25
γ	Carol	632	θ	λ	σ	25
δ	Dan	742	η	μ	τ	30
ϵ	Eve	893	θ	ν	υ	30
ζ	Adam	897	η	ξ	ϕ	30

Rating: order-preserving ciphertexts.
Gender: distinguishable ciphertexts.
Salary and ID: indistinguishable ciphertexts.

simplicity, we keep the definition of probabilistic record protection vague in this paper because the limit of obtainable information can depend on the background knowledge of the attacker, the number of outsourced records and the specific CPI approach. This protection level can be subdivided further in future work in order to cope with this variety.

Example: Assume that the combination of the attributes *name*/*salary* of a dataset is considered confidential, i.e., it must be impossible for an attacker to map a person's salary to their name. In Table 2, the attacker cannot discover any salary because it is encrypted for each record. Computational record protection is ensured because the attacker only learns something about the confidential attribute combination *name*/*salary* if they know the encryption key for the encrypted *salary* values.

In Table 2, the attacker learns something about the confidential attribute combination *name*/*age*. For instance, the attacker learns that Bob's age is either 23, 25 or 30. Thus, only probabilistic record protection is ensured.

Assumptions on attackers

Our attacker taxonomy is shown in Figure 2. The CPI approaches presented in this paper make the basic assumption that an attacker is **honest-but-curious**, i.e., the attacker observes the outsourced data and/or queries but behaves according to protocol. In other words, the attacker does not manipulate the data or query results. Additional integrity-preserving approaches can be utilized to enable detection of malicious behavior before data confidentiality is compromised. However, these approaches are beyond the scope of this paper. Most approaches assume that the data is outsourced to a single SP. Others assume that the data is outsourced to multiple, non-colluding SPs. With non-colluding SPs, only a single SP can be compromised by an attacker and the outsourced data can be split up between multiple (non-colluding) SPs to protect it. Notice that approaches that protect against attackers who are able to compromise multiple SPs also protect against attackers who are just able to compromise a single SP.

In the following, we propose an attacker taxonomy at a level of detail that we deem suitable for real-world application. In our opinion, a finer-grained model would provide only limited additional benefit as users typically possess limited knowledge about the attacker and are hence unable to provide a more detailed model. On the other hand, a coarser-grained model would not allow the description and harnessing of key differences between individual CPIs.

The existing approaches also make different assumptions with respect to what an attacker is able to observe. While some approaches assume that an attacker is only able to view **the outsourced data** (D), others assume they can **observe modifications** (M) to the data or even **queries** (Q) that are executed on the outsourced data. In particular, to be able to monitor queries on the data, an attacker has to have control over an SP when the queries are executed. An attacker who only compromises the SP for a very short period of time is able to take a snapshot of the outsourced data, but cannot observe query executions. An attacker, who has compromised an SP multiple times for a short period of time can analyze the differences between the two data versions observed and infer the modifications that were performed. Furthermore, if the data modifications are logged by the SP for versioning purposes, they can even be observed by attackers who compromise an SP just once. Notice that approaches that protect against attackers who are able to observe queries (Q-Attackers) also protect against attackers who are able to observe modifications (M-Attackers) because performing a modification requires a query to the SP.

Another important aspect in this regard is the assumed background knowledge (BK) of the attacker, i.e., the knowledge that the attacker has in addition to the outsourced data or monitored queries/modifications. We differentiate between approaches that assume attackers have: **no background knowledge** (NBK), **background knowledge of the outsourced data's schema** (BKS), **background knowledge of the outsourced data's content** (BKD) and **background knowledge of the outsourced data and executed queries** (BKQ). While a BKS-attacker knows the data's schema, i.e., the data types of an attribute and its value range, they do not know anything about the specific outsourced data. A BKD-attacker has specific knowledge of the outsourced data such as the frequency distribution of the attribute values. We argue that an attacker is more likely to possess BKS knowledge rather than specific BKD knowledge of the data. Also notice that it can be possible to infer BKS knowledge from BKD knowledge as we show in the following example. Thus, an approach that protects against BKD-attackers also needs to protect against BKS-attackers.

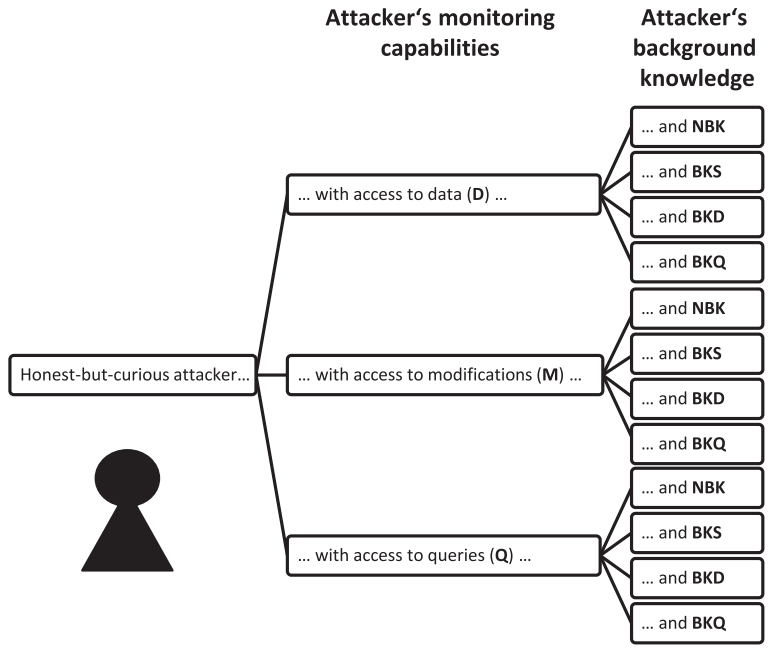

Figure 2 Attacker taxonomy.

Example: The knowledge that marks are integer values in the range 1-6 is considered BKS knowledge. The frequency distribution of the outsourced marks constitutes BKD knowledge. Based on the frequency distribution, it may also be possible to derive the BKS knowledge that marks are generally within the 1-6 value range.

A BKQ-attacker has knowledge of both the outsourced data and queries that are executed.

Example: The knowledge that the record corresponding to the director of a company is the one queried most often constitutes BKQ knowledge. An attacker who is able to observe queries can use this knowledge to identify the record of the director.

It should be noted that in the real world, the attacker's capabilities are bound to change over time, for instance when the attacker gains additional background knowledge. Thus, it has to be checked continuously whether new threats arise. If that is the case, the attacker model has to be adapted and the CPI choice has to be reassessed and adapted if necessary.

Methodology to assess CPIs

To map CPIs to deployment scenarios in which they are applicable, they have to be assessed with regard to the taxonomy requirements they are able to satisfy. CPIs can be assessed based on the presented taxonomy as follows: In most cases, the functionality and the protection level of a CPI is clearly specified by the authors who published the CPI. Furthermore, most authors provide *security properties* for their CPIs that declare what an attacker who can monitor the outsourced (encrypted) data and the executed queries is able to observe. A mapping that maps security properties onto the attackers is not always provided. However, such a mapping is crucial to compare the protection guarantees of CPIs.

Whether confidentiality is preserved by using a CPI approach depends on both the security properties of the approach and the attacker model assumed. In this section, we show which security properties are needed to protect against various attacker models. We differentiate between the following security properties:

- **Content Confidentiality**: The content confidentiality provided by an approach reflects how the protected attribute values are represented at the SP. We differentiate between **order-preserving ciphertexts**, **distinguishable ciphertexts**, and **indistinguishable ciphertexts**. Order-preserving ciphertexts leak information on the ordering of the corresponding plaintext values to the SP. If distinguishable ciphertexts are used, the SP is able to differentiate between ciphertexts that encrypt value A from those that encrypt value B. In particular, deterministic encryption schemes produce distinguishable ciphertexts because encrypting equal plaintext values results in equal ciphertexts. If indistinguishable ciphertexts are outsourced, the SP is not able to differentiate ciphertexts that encrypt value A from those that encrypt value B. In particular, this implies that encrypting a plaintext value twice

results in different ciphertexts. Furthermore, based on indistinguishable ciphertexts, attackers can not discover which values are contained in the outsourced data and which are not.

- **Access Confidentiality**: An approach that provides access confidentiality ensures that based on the query results that the SP transmits to the mediator it is not possible to infer which records are affected by the query with regard to the protected attribute (e.g., which records are targeted by a SELECT query's selection condition on the protected attribute or which protected attributes have been newly inserted by an INSERT query).
- **Pattern Confidentiality**: An approach that provides pattern confidentiality ensures that an attacker is not able to observe query patterns, i.e., it is not possible for an attacker to observe that two distinct queries are targeting the same set of records [9].

The security properties that are necessary to protect against specific attacker models are shown in Figure 3 and explained below. Notice that Figure 3 is based on the worst case assumptions concerning the observable data/modifications/queries and background knowledge of the attacker. For instance, if an attacker is able to observe modifications, we assume that they can observe *arbitrary* insert, update and delete operations even if they would never occur in certain deployment scenarios, e.g., where records are never updated or deleted. With regard to the CPI approaches that we introduce, we show that those that do not have the necessary security properties given in Figure 3 can provide protection if these worst case assumptions are restricted (e.g., update/delete operations are not performed, therefore they are not observable).

If the attacker has no background knowledge (NBK), outsourcing order-preserving ciphertexts is sufficient. By analyzing these ciphertexts, the attacker learns the ordering of the attribute values, however since they have no background knowledge they are not able to discover any confidential plaintext values.

Example: Based on the order-preserving ciphertexts of *rating* in Table 2, an attacker can observe that Bob has a better *rating* than Adam. However, without background knowledge on how the ratings are specified, discovering the plaintext rating of Adam and Bob is not possible.

If the attacker has background knowledge of the data schema (BKS), order-preserving ciphertexts may no longer provide protection. CPI approaches that outsource distinguishable ciphertexts can be used to protect against an attacker with BKS knowledge because distinguishable ciphertexts do not contain any order information, which means that the value range of an attribute cannot be used to map distinguishable ciphertexts onto plaintext values.

Example: Based on the order-preserving ciphertexts of *rating* in Table 2 and the knowledge that the ratings correspond to integer values between 1 and 6, an attacker can infer that the smallest of the six ciphertexts (345) has to correspond to the plaintext rating 1. However, based on the knowledge that *gender* can take the values "m" or "f", an attacker is not able to derive whether η corresponds to "m" or "f" from Table 2.

However, once the attacker can observe range queries and range query results, they may again be able to infer the order of distinguishable ciphertexts. When the order of the ciphertexts is known, BKS knowledge can be applied to reveal confidential attribute values. Therefore, in order to protect against an attacker that can observe range queries and has BKS knowledge, an approach needs to provide access confidentiality to hide the query results. The same holds true for attackers that are able to observe modifications because a modification often implies the evaluation of a query.

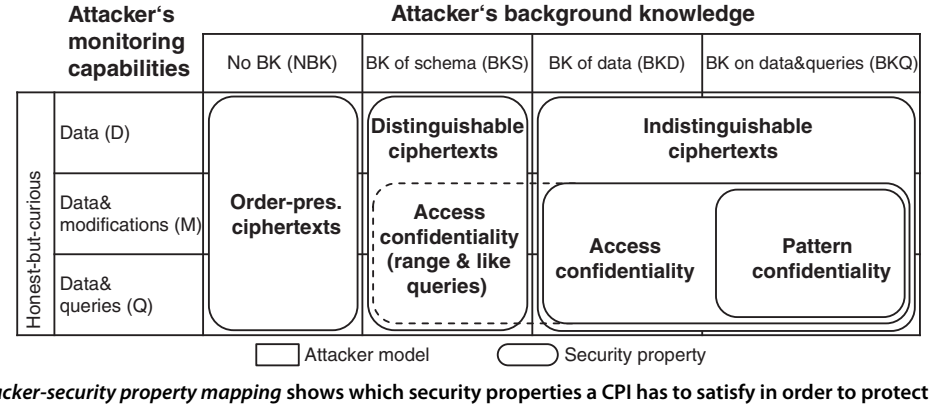

Figure 3 The *attacker-security property mapping* shows which security properties a CPI has to satisfy in order to protect against specific attackers. An approach that provides computational record protection against a given attacker model has to satisfy the security properties that cover the cell that represents the assumed attacker model.

Example: If an attacker observes the results of two range queries containing distinguishable ciphertexts $\{\omega, \psi\}$ and $\{\psi, \varphi\}$, they can infer that either $\omega < \psi < \varphi$ or $\varphi > \psi > \omega$ holds true and that there is no other value between ω and ψ or ψ and φ.

If the attacker has background knowledge of the data (BKD), they can reveal plaintext values based on distinguishable ciphertexts as we show in the following example. To protect against attackers that can only observe outsourced ciphertexts and that have BKD knowledge, an approach can only be considered secure if the outsourced ciphertexts are indistinguishable with regard to their content. If the ciphertexts are indistinguishable, the attacker has no way of correlating the BKD knowledge with the outsourced data.

Example: If the attacker knows that "m" is the most frequent *gender* in the data, this plaintext value can be mapped to the most frequent distinguishable ciphertext η in Table 2. If the attacker knows that 20000 is the most frequent *salary*, they cannot use this knowledge on Table 2 because the ciphertexts of *salary* are not distinguishable.

However, once the attacker can observe queries and query results, it may once again be possible to distinguish the otherwise indistinguishable ciphertexts. Once the ciphertexts are distinguishable, BKD knowledge can be applied to reveal the confidential attribute values. Therefore, in order to protect against an attacker that can observe queries and has BKD knowledge, an approach needs to provide access confidentiality to hide the query results. The same holds true for attackers that can observe modifications because a modification often implies the evaluation of a query.

Example: If an attacker observes that records 1, 2 and 4 are the result of a query that was evaluated on the indistinguishable attribute values of *salary* in Table 2, the attacker could infer that records 1, 2 and 4 have the same *salary*. Combined with the knowledge that Adam earns 20000, the attacker could then reveal the salary of Bob and Dan. Furthermore, if an attacker observes that records 1, 2 and 4 are deleted by one DELETE query that filtered on *salary* (`DELETE ...WHERE salary=?`), it can infer that the deleted records had the same salary.

An approach that can be used to protect data from attackers who are able to monitor queries, and have background knowledge of the executed queries (BKQ), has to offer pattern confidentiality. Otherwise, an attacker would be able to observe the query patterns and correlate the background knowledge on the queries to gain confidential information.

Example: If an attacker knows that the records with a *salary* of between 25000 and 40000 are queried most often, and observes that the majority of queries return the set of records 5 and 6, they learn that these records have a salary of between 250000 and 40000.

Survey of existing CPIs

In the following, we survey the most popular CPIs, map them to deployment scenarios in which they are applicable and provide an overview of their performance. Using the proposed methodology, we build a CPI catalog that summarizes our analysis by showing how the existing CPIs map to the deployment scenario requirements of our proposed taxonomy. The CPI catalog lists the requirements that a CPI is able to satisfy, and allows a quick determination of which CPIs match the requirements of a specific deployment scenario in terms of the assumed attacker model, the required CPI functionality, and the protection level.

CPI catalog

The CPI catalog that contains different classes of CPI approaches can be found in Table 3. Depending on the functionality that is utilized, some approaches protect against multiple attacker models. In the following, we provide a brief description and explain the categorization of each CPI approach.

Deterministic indexes

In order for the SP to evaluate equality selections on attribute values without revealing the values, each plaintext value can be mapped to a deterministic substitute. For instance, these substitutes can be keyed hash values of the plaintext value or ciphertexts that are produced by deterministic encryption schemes. Both the keyed hash function and the deterministic encryption schemes ensure that mapping single deterministic substitutes back to the plaintext value without knowing the key is infeasible. In order to search for a certain attribute value a, the attribute value a is replaced by its deterministic substitute within the query.

Example: To evaluate the query `SELECT ...WHERE Gender=m` on Table 2, the attribute value "m" is first mapped to its deterministic substitute η by the mediator using the secret key. The query `SELECT ...WHERE Gender=`η can then be passed to the SP, which returns record 1, 2, 4 and 6 as the result.

Since deterministic ciphertexts are distinguishable and only equality selections are evaluated, the approach protects against Q&BKS-attackers, i.e., attackers with background knowledge of the data's schema (cf. Figure 3).

In order to also use deterministic indexes against attackers with background knowledge of the data, the *flattened hash indexes* approach was proposed [10]. The basic idea of flattened hash indexes is to map different plaintext values to the same deterministic substitute in such a way that each deterministic substitute occurs the same number of

Table 3 CPI catalog

CPI approach	Satisfiable deployment requirements															CPI security properties
	Functionality							Prot. level	Attacker model							
	Queries				Modification				Monitoring cap.			Knowledge				(Properties in brackets are only provided to a certain degree.)
	ES	RS	LS	AG	Insert	Update	Delete		Data	Mod.	Quer.	NBK	BKS	BKD	BKQ	
Deterministic Indexes	X				X	X	X	C	✓	✓	✓	✓	✓			Distinguishable ciphertexts
Deterministic Indexes (Flattened) [10]	X				X	X	X	P	✓			✓	✓	✓	✓	(Indistinguishable ciphertexts)
Bucketization [11,12]		X			X	X	X	C	✓			✓	✓			Distinguishable ciphertexts
		X			X	X	X	C	✓	✓	✓	✓				
		X			X			C	✓	✓		✓	✓			
Bucketization (Flattened) [11,12]		X			X	X	X	P	✓			✓	✓	✓	✓	(Indistinguishable ciphertexts)
Order-Preserving Encryption [13,14]	X	X			X	X	X	C	✓	✓	✓	✓				Order-preserving ciphertexts
Searchable Encryption [15-20]	X	X	X		X	X	X	C	✓			✓	✓	✓	✓	Indistinguishable ciphertexts
	X	X	X		X	X	X	C	✓	✓	✓	✓				
	X				X	X	X	C	✓	✓	✓	✓	✓			
	X				X			C	✓	✓		✓	✓	✓		
Encrypted B-Trees [10,22]	X	X	x[a]		X	X	X	C	✓			✓	✓	✓	✓	Indistinguishable ciphertexts
	X	X	x[a]		X	X	X	C	✓	✓	✓	✓				
	X	X	x[a]		X			C	✓	✓		✓	✓	✓		
Encrypted B-Trees (Shuffled) [24]	X	X	x[a]		X	X	X	P	✓	✓	✓	✓	✓	✓	✓	Indistinguishable ciphertexts (Access & pattern confidentiality)
Fragmentation [25]	X	X	X	X	X	X	X	P	✓			✓	✓	✓	✓	(Indistinguishable ciphertexts)
Fragmentation (Non-colluding SPs) [26-28]	X	X	X	X	X	X	X	C	✓	✓	✓	✓	✓	✓		Indistinguishable ciphertexts Access confidentiality
Homomorphic Encryption [29-33]				X	X	X	X	C	✓	✓	✓	✓	✓	✓	✓	Indistinguishable ciphertexts Access & pattern confidentiality
Oblivious RAM [34,35]	X	X	x[a]		X	X	X	C	✓	✓	✓	✓	✓	✓	✓	Indistinguishable ciphertexts
Oblivious RAM (Non-colluding SPs) [36]	X	X	x[a]		X	X	X	C	✓	✓	✓	✓	✓	✓	✓	Access & pattern confidentiality
Private Information Retrieval [39-42]	X	X	x[a]					C	✓	✓	✓	✓	✓	✓	✓	Indistinguishable ciphertexts
Private Information Retrieval (Non-colluding SPs) [37,43,44]	X	X	x[a]					C	✓	✓	✓	✓	✓	✓	✓	Access & pattern confidentiality

[a]Like selections are supported to a limited degree (e.g., prefix matching).

Legend

ES: Equality selection; NBK: No background knowledge; C: Computational record protection RS: Range selection; BKS: Background knowledge of the data's schema P: Probabilistic record protection; LS: Like selection; BKD: Background knowledge of the data's content; AG: Aggregation; BKQ: Background knowledge of the data's content and queries.

times. By doing this, the deterministic ciphertexts become indistinguishable in the sense that background knowledge on the frequency distribution of an attribute can no longer be applied. However, flattening the distribution of the deterministic substitutes does not entirely prevent background knowledge from being applied on single plaintext records because equal plaintext values still map to equal deterministic substitutes [10].

Example: Even if Table 2 contained as many female as male persons, an attacker who knows that Adam has *gender* "m" could infer that Carol cannot have *gender* "m" because Carol's record contains θ, which does not match Adam's η.

Thus, flattened hash indexes can be used against D&BKQ-attackers, but only provide probabilistic database protection because an attacker with background knowledge may gain information about some of the plaintext values based on the outsourced data.

Bucketization

Range selections cannot be evaluated based on deterministic indexes because deterministic ciphertexts do not maintain the order of the plaintext values. The idea of value bucketization [11,12] addresses this by sorting the plaintext values into buckets, i.e., continuous value ranges. For each plaintext value, only the corresponding bucket ID that does not contain information on the content of the bucket is outsourced. In order to query for a range of values, the ID of each contained and intersecting bucket is queried.

Since the outsourced values are distinguishable ciphertexts, bucketization can be considered secure against a D&BKS-attacker that is only able to access the data and has background knowledge of the data's schema. However, once an attacker can observe queries or modifications that update/delete records, which are selected based on a bucketization index, it can deduce the ordering of the bucket IDs and thus transform distinguishable ciphertexts into order-preserving ciphertexts. Therefore, if queries can be monitored by the attacker, bucketization is only secure against Q&NBK-attackers that have no background knowledge.

Like deterministic indexes, buckets can be flattened so that each bucket contains the same number of records. For the same reasons as flattened hash indexes, flattened bucketization can be used against D&BKQ-attackers, however it only provides probabilistic database protection.

Order-preserving encryption

To allow the SP to evaluate range selections on encrypted attribute values, order-preserving encryption schemes (OPES) can be used to encrypt the attribute values [13,14]. OPES schemes maintain order when mapping plaintext values to ciphertext values, i.e., the ciphertext values have the same order as the corresponding plaintext values. Thus, it is possible for the SP to evaluate $<$ and $>$ operators without decrypting the ciphertexts and revealing the plaintext values.

Since order-preserving ciphertexts are outsourced, the approach only protects against Q&NBK-attackers without background knowledge (cf. Figure 3).

Searchable encryption

Searchable encryption schemes [15-17,17-20] can be used to encrypt data values in such a way that the SP can check whether a ciphertext contains a value that matches a given predicate or not. The outsourced ciphertexts are indistinguishable. Searchable encryption schemes enable the mediator to generate a token based on the predicate that has to be checked and the secret key that was used to encrypt the data. This token is passed to the SP. While it is not possible to determine the predicate to be evaluated from the token without the secret key, the token can be used by the SP to check which ciphertexts match the (unknown) predicate. Predicates can be utilized to encode equality, range and like selections.

Since the outsourced ciphertexts are indistinguishable, searchable encryption can be used to protect the data from D&BKQ-attackers that have access to the outsourced data as well as background knowledge on the data and the queries. However, once an attacker can monitor queries or modifications it is not necessary for them to have background knowledge of the data because indistinguishable ciphertexts alone do not protect against Q&BKD-attackers and searchable encryption approaches do not provide access confidentiality (cf. Figure 3) [21]. Furthermore, if range selections can be monitored, it is not necessary for the attacker to have background knowledge because they can establish an order between the ciphertext values based on the monitored range queries, as was the case with bucketization.

Encrypted B-Trees

In traditional database management systems, B-Trees are used to create indexes and speed up query execution. To enable efficient query execution based on indistinguishable ciphertexts stored by the SP, encrypted B-Trees containing only indistinguishably encrypted nodes were proposed [10,22]. Since the encrypted nodes of the encrypted B-Tree are indistinguishable for the SP, the trustworthy mediator has to maintain the encrypted B-Tree and participate in the execution of queries. To retrieve a leaf that references a record with a certain attribute value, the mediator first retrieves the root node of the B-Tree, decrypts it and selects the node for descending into the tree as in the unencrypted case. This node is then retrieved once again and the process is repeated until the

target leaf node is reached. Thus, $log(n)$ communication rounds are required to retrieve a record.

Since the outsourced ciphertexts are indistinguishable, encrypted B-Trees can be used to protect attributes against D&BKQ-attackers that have access to the outsourced data as well as background knowledge on the data and the queries (cf. Figure 3). However, since encrypted B-Trees do not provide access confidentiality, they are not suited to provide protection against a Q&BKD-attacker that has background knowledge of the data and can monitor queries/modifications. Once an attacker can observe queries or modifications, they are not only able to distinguish ciphertexts, but can also infer the order of the ciphertexts due to the B-Tree's ordered structure [23]. Thus, if queries or modifications that target specific records can be observed, encrypted B-Trees can only provide security guarantees against Q&NBK-attackers that have no background knowledge.

In order to make encrypted B-Trees applicable to stronger attacker models, shuffling the B-Tree after every query was proposed to achieve access and pattern confidentiality [24]. For each value looked up in the B-Tree, e different values are looked up so that e nodes need to be retrieved at each stage of the B-Tree. Once the leaf nodes for all the cover searches are received, the nodes of each stage are shuffled by the client and written back to the B-Tree at the SP. Thus, the access patterns of two identical queries look different to the SP. However, since only e nodes are shuffled, the SP is still able to distinguish queries with a certain probability. This amount of information decays with the number of queries that are executed in between the queries of interest. It may be possible for the attacker to apply background knowledge concerning the access pattern with a certain probability. Thus, shuffled B-Trees only guarantee *probabilistic database protection* against Q&BKQ-attackers that have background knowledge of query patterns.

Fragmentation

In some cases, it is not attribute values, but attribute value combinations that are considered confidential. Fragmentation approaches split relations up into multiple fragments to protect the attribute combinations. For instance, even though an attacker is allowed to see plaintext attribute values of attribute *age* and *name*, it must be impossible to link any age to a name. In order to achieve that, the relation can be split up into two unlinkable fragments, where each one contains either *name* or *age* as shown in Table 2. Since no encryption is applied, the SP can still evaluate equality, range and like selections as well as aggregations.

To protect a record, it suffices to protect one of the attributes that should not be linked by not storing the attribute in the same fragment. As the SP does not learn the values of attributes that are not part of a fragment, the values of such attributes can be considered indistinguishable ciphertexts.

The fragments can all be stored on a single SP or distributed on multiple non-colluding SPs. If the fragments are stored on a single SP [25], the confidentiality of attribute combinations can only be guaranteed against D&BKQ-attackers that are not able to observe any modifications of the data. For instance, in order to insert a new record, partial records are inserted into each fragment. An attacker that observed the newly inserted partial records can infer that they belong to the same record and link them. As shown in Figure 3, to protect against stronger attackers, access confidentiality is necessary to obfuscate which records were inserted. Notice that storing fragments on a single SP can only provide *probabilistic record protection*: For instance, in Table 2 the attacker learns that Bob's age is either 23, 25 or 30 based on the outsourced data. Thus, values of protected attributes are only somewhat indistinguishable.

Storing the fragments on multiple non-colluding SPs [26-28] can be considered secure against Q&BKD-attackers that are able to observe queries and modifications. Since the fragments are stored on different SPs, an attacker that compromised a single SP is not thereby enabled to observe which partial records are inserted into each fragment or to join them to reveal the confidential attribute combination. Furthermore, *computational record protection* can be achieved because the value range of the protected attributes cannot be narrowed down by the SPs as only a single fragment can be observed by each SP. Thus, values of protected attributes are truly indistinguishable. For instance, an attacker that has access to the fragment that contains *age* in Table 2 can infer that the age of a person in the dataset is either 23, 25 or 30. However, since the attacker has no access to the other fragment, they do not learn any names that are contained in the database and can be possibly linked to the age.

Homomorphic encryption

Homomorphic encryption schemes [29-31] can be used to produce ciphertexts that can be aggregated without knowing the secret key used for encryption [32,33]. The execution of certain operations in the ciphertext domain has the effect of executing an operation such as addition or multiplication is performed in the plaintext domain. Thus, the mediator can outsource homomorphically encrypted ciphertexts that can be aggregated by the SP (e.g., summed, depending on the scheme utilized).

The ciphertexts produced by homomorphic encryption schemes can be considered indistinguishable for an attacker. Aggregation queries target all records, rather than specific ones, and therefore naturally ensure access and pattern confidentiality. More precisely, all the records

that were selected are either based on other indexing approaches such as deterministic indexes or B-Trees. Thus, homomorphic encryption can be considered secure against strong Q&BKQ-attackers with background knowledge of the data and queries and the ability to monitor queries.

Oblivious RAM and private information retrieval

The goal of Oblivious RAM (ORAM) [34-36] approaches is to ensure that queries evaluated on the data are indistinguishable and that the SP does not know which records are returned or whether the same query has been executed before. ORAM approaches shuffle the outsourced encrypted data structure with each data access to ensure that the executions of multiple identical/similar queries look entirely different and cannot be distinguished by the SP. The property of indistinguishable queries can only be guaranteed if there is access and pattern confidentiality. Thus, ORAM approaches ensure access and pattern confidentiality and can be considered secure against Q&BKD-attackers.

Private Information Retrieval (PIR) [37,38] approaches obfuscate data access patterns and can be considered secure against Q&BKD-attackers. Unlike ORAM, PIR can only be applied for data retrieval, not for writing data, which is a common requirement in DaaS scenarios. In contrast to ORAM, PIR approaches can obfuscate query access patterns in a single round of communication. Computational PIR approaches [39-42] achieve this at the expense of increased computational cost for the SPs, while information-theoretical PIR approaches [37,43,44] obfuscate access patterns based on non-colluding SPs. To evaluate queries (e.g., range selections) based on probabilistic ciphertexts, PIR approaches can be combined with methods such as encrypted B+ trees. This implies a logarithmic number of communication rounds, which would cancel out the benefits of PIR over ORAM in this regard.

CPI efficiency

In order to facilitate assessment of whether a CPI is suited for a given deployment scenario, we also provide a high level overview of the query execution performance achieved by each CPI. While a fine-grained performance evaluation is beyond the scope of this article, our aim is to give a rough estimate of the most important performance metrics. We distinguish between *transmission overhead*, which we define as the amount of data transmitted during a query, the number of sequential *communication rounds* between the mediator and the SP to answer a query and the number of *entries* that need to be touched by the SP in order to evaluate a query. Lastly, we categorize each CPI with respect to the *computational overhead* induced for the mediator and the SP according to four levels. We assign the level *none* if no or almost no calculations need to be performed. Overhead is assumed to be *low* if only lightweight cryptographic operations such as hashing are performed. If only symmetric encryption schemes are used, we assume the overhead to be *moderate*. If more resource-consuming asymmetric encryption schemes are utilized, we regard the computational overhead as *high*.

In the following, some of the most important findings are highlighted:

Compared to other CPI approaches, deterministic indexes and fragmentation induce the lowest overhead. With fragmentation, however, the SP is not able to fully execute a query based on a single fragment that does not contain all the attributes that are relevant for the query. Thus, a large number e of false-positive records may be unnecessarily transmitted and have an impact on query execution performance. In the non-colluding-SP model, this is less of a problem because attributes can be stored redundantly in multiple fragments.

The performance of flattened deterministic indexes and bucketization highly is highly dependent on the distribution of the outsourced attribute values. In unequally distributed datasets, many different plaintext values need to be mapped to a deterministic substitute in order to ensure a flat index. Querying for one of these plaintexts will result in the unnecessary transmission of many false-positive records e.

B-Trees require log(n) consecutive communication rounds between the mediator and the SP to retrieve the tree nodes. This leads to a stacking of network latency between the mediator and SP during query execution.

Using searchable encryption, the SP has to scan all outsourced ciphertexts contained in the index to find the records that match a query.

Homomorphic encryption induces a comparatively high computational overhead for the SP to aggregate ciphertexts and for the mediator to encrypt/decrypt the ciphertexts. We refer to partially homomorphic encryption here; the overhead for fully homomorphic encryption is much higher.

ORAM and PIR approaches each have their own trade-offs between transmission overhead, the necessary communication rounds and computational overhead. Overall they are considered expensive compared to other approaches that protect against weaker attacker models.

Comparative analysis and discussion

The proposed CPI assessment with regard to the CPIs' functionality, their protection guarantees against various attacker models and their efficiency allows recommendations to be made on which CPIs should be used in which context. Figure 4 illustrates these recommendations under the assumption that computational record protection is required. For each given attacker model, the figure lists

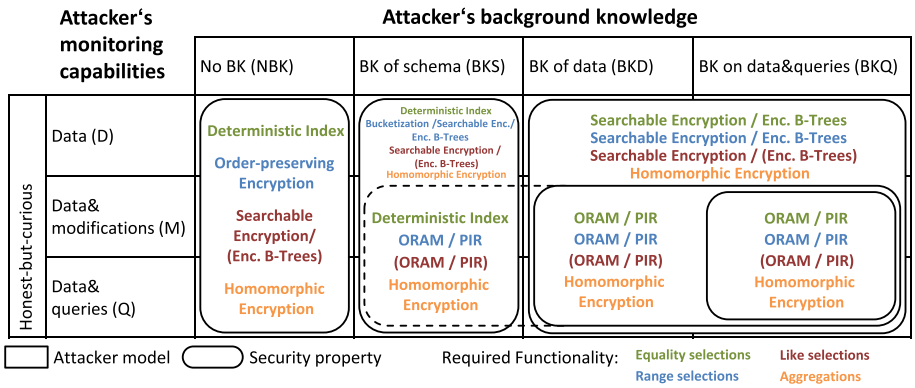

Figure 4 Efficiency-optimal CPIs that offer the required functionality and the security properties that are required to provide computational record protection against each attacker model (fragmentation excluded).

the most efficient CPIs that can be applied without threatening data confidentiality. In some cases multiple CPI candidates are listed as it depends on the deployment scenario and its requirements which CPI can be considered the most efficient. For instance, searchable encryption and encrypted B-Trees can be used to evaluate range selections and provide similar protection guarantees. As shown in Table 4, searchable encryption induces more computational overhead and requires the SP to touch more entries than encrypted B-Trees, whereas encrypted B-Trees require multiple rounds of communication. Thus, searchable encryption should be favored over encrypted B-Trees if network latency is expected to be high. To provide computational record protection, fragmentation alone is typically insufficient as too few SPs are available. We omitted the fragmentation approach in the figure. However, it can be applied in addition to other CPIs to avoid having to apply more expensive CPIs [7].

Table 4 Performance categorization of existing approaches

	Transmission overhead	Communication rounds	Touched entries	Computation	
				SP	Mediator
Deterministic Indexes	O(k)	1	O(k)	none	low
Deterministic Indexes (Flattened)	O(k+e)	1	O(k+e)	none	low
Bucketization	O(k+e)	1	O(k+e)	none	low
Bucketization (Flattened)	O(k+e)	1	O(k+e)	none	low
Order-Preserving Encryption	O(k)	1	O(k)	none	low
Searchable Encryption	O(k)	1	O(n)	moderate	moderate
Encrypted B-Trees	O(log(n)+k)	log(n)	O(log(n)+k)	none	moderate
Encrypted B-Trees (Shuffled)	O(2∗e∗log(n)+k)	log(n)+1	O(2∗e∗log(n)+k)	none	moderate
Fragmentation	O(k+e)	1	O(k+e)	none	none
Fragmentation (non-colluding SPs)	O(k)	1	O(k)	none	none
Homomorphic Encryption	O(1)	1	O(k)	high	high
Oblivious RAM	Depends on the specific approach				
Example [35]	$O(\log(n)^2 + \log(n)*k)$	log(n)+1	$O(\log(n)^2+\log(n)*k)$	none	moderate
Example [36] (non-colluding SPs)	SP→Client: O(log(n)+k) SP→SP: $O(\log(n)^2+\log(n)*k)$	log(n)+1	$O(\log(n)^2+\log(n)*k)$	low	low
Private Information Retrieval	Depends on the specific approach				
Example [38,44] (non-colluding SPs)	Hash Index: $O(n^{\frac{1}{3}})+k$ B-Tree: $O(\log(n)*n^{\frac{1}{3}}+k)$	1 log(n)	$O(n^{\frac{1}{3}})+k$ $O(\log(n)*n^{\frac{1}{3}}+k)$	low	low

Legend
n: Number of outsourced attribute values.
k: Number of records matching a query.
e: Number of (unnecessarily transmitted) false positive records.

Key conclusions and future work

The survey shows that while no universal CPI approach exists that supports every query type and protects the data from all attacker models, the existing CPIs can cover many deployment scenarios. The requirement triad of *functionality*, *attacker model* and *protection level* is important in terms of assessing the applicability of an approach for a given deployment scenario. These requirements are interdependent, i.e., in some cases a specific CPI can protect against a stronger attacker model if the user is willing to accept less functionality or less protection. Furthermore, our study clearly indicates that CPI approaches that protect against strong attacker models induce a higher performance overhead than approaches that protect against weak attackers.

In the future, some of the important tasks of the database-as-a-service community will be improving CPI performance as well as leveraging individual CPI benefits by integrating them into frameworks that automate the choice of CPIs. Approaches for integrating CPIs have been proposed [7,8,45], however, further exploration of the potential of the symbiotic effects between CPIs with regard to security and performance is another interesting future research challenge.

Abbreviations

CPI: confidentiality preserving index; SP: storage provider; ES: equality selection; RS: range selection; LS: like selection; AG: aggregation; D-attacker: attacker that can observe outsourced data; M-attacker: attacker that can observe modifications on outsourced data; Q-attacker: attacker that can observe queries on the outsourced data; NBK: no background knowledge; BKS: background knowledge on the data's schema; BKD: background knowledge on the data; BKQ: background knowledge on queries; C: computational record protection; P: probabilistic record protection.

Competing interests

The authors declare that they have no competing interests.

Authors' contributions

JK developed the taxonomy and methodology, carried out the study on CPIs and drafted the manuscript. KJ served as a discussion partner and revised the manuscript. HH provided advice and reviewed/revised the manuscript. All authors read and approved the final manuscript.

Authors' information

Jens Köhler received his diploma degree in computer science in 2011 from the department of computer science of the Karlsruhe Institute of Technology. Since 2012, he has been a researcher at the Institute for Telematics and the Steinbuch Centre for Computing (SCC) in the Decentralized Systems and Network Services Research Group (DSN) and is pursuing a Ph.D. degree. His work focusses on IT security management and confidential data outsourcing as well as federated identity management in particular.

Konrad Jünemann studied computer science at the Karlsruhe Institute of Technology (KIT). In 2008, he worked for SAP Corporate Research Brisbane. Since 2009 he has been a researcher at the Institute for Telematics and the Steinbuch Centre for Computing (SCC) in the research group of Prof. Dr. Hartenstein and is pursuing a Ph.D. degree. His work focusses on confidential data outsourcing, IT security management and the analysis, modeling, and simulation of decentralized systems.

Hannes Hartenstein is a full professor of computer science and a director of the Steinbuch Centre for Computing at the Karlsruhe Institute of Technology, Karlsruhe, Germany. He received a diploma degree in mathematics and a Ph.D. degree in computer science from Albert Ludwigs University, Freiburg, Germany. Prior to joining KIT, he was a senior research staff member at NEC Europe. His research interests include mobile networks, virtual networks, security, and information technology management. He is a member of the scientific directorate of the Leibniz Centre for Informatics, Schloss Dagstuhl.

Received: 24 July 2014 Accepted: 14 November 2014
Published online: 31 January 2015

References

1. Huang J, Nicol D (2013) Trust mechanisms for cloud computing. Journal of Cloud Computing 2(1):1–14
2. Gonzalez N, Miers C, Redígolo F, Simplício M, Carvalho T, Nälund M, Pourzandi M (2012) A quantitative analysis of current security concerns and solutions for cloud computing. Journal of Cloud Computing 1(1):1–18
3. Wei DSL, Murugesan S, Kuo S.-Y, Naik K, Krizanc D (2013) Enhancing data integrity and privacy in the cloud: An agenda. Computer 46(11):87–90
4. Mykletun E, Narasimha M, Tsudik G (2006) Authentication and integrity in outsourced databases. ACM Transactions on Storage 2(2):107–138
5. Juels A, Kaliski BS Jr (2007) Pors: Proofs of retrievability for large files. In: Proceedings of the 14th ACM Conference on Computer and Communications Security. CCS'07. ACM, New York, NY, USA. pp 584–597
6. Damiani E, di Vimercati SDC, Foresti S, Jajodia S, Paraboschi S, Samarati P (2005) Key management for multi-user encrypted databases. In: Proceedings of the 2005 ACM Workshop on Storage Security and Survivability. StorageSS '05. ACM, New York, NY, USA. pp 74–83
7. Köler J, Jünemann K (2014) Securus: From confidentiality and access requirements to data outsourcing solutions. In: Hansen M, Hoepman J-H, Leenes R, Whitehouse D (eds). Privacy and Identity Management for Emerging Services and Technologies. IFIP Advances in Information and Communication Technology, vol. 421. Springer, Berlin Heidelberg. pp 139–149
8. Popa RA, Redfield CMS, Zeldovich N, Balakrishnan H (2011) Cryptdb: Protecting confidentiality with encrypted query processing. In: Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles. SOSP '11. ACM, New York, NY, USA. pp 85–100
9. De Capitani di Vimercati S, Foresti S, Paraboschi S, Pelosi G, Samarati P (2011) Efficient and private access to outsourced data. In: Proceedings of the 2011 31st International Conference on Distributed Computing Systems. ICDCS '11. IEEE Computer Society, Washington, DC, USA. pp 710–719
10. Ceselli A, Damiani E, Vimercati SDCD, Jajodia S, Paraboschi S, Samarati P (2005) Modeling and assessing inference exposure in encrypted databases. ACM Transactions on Information and System Security 8(1):119–152
11. Hacigümüş H, Iyer B, Li C, Mehrotra S (2002) Executing sql over encrypted data in the database-service-provider model. In: Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data. SIGMOD '02. ACM, New York, NY, USA. pp 216–227
12. Hore B, Mehrotra S, Canim M, Kantarcioglu M (2012) Secure multidimensional range queries over outsourced data. VLDB Journal 21(3):333–358
13. Agrawal R, Kiernan J, Srikant R, Xu Y (2004) Order preserving encryption for numeric data. In: Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data. SIGMOD '04. ACM, New York, NY, USA. pp 563–574
14. Boldyreva A, Chenette N, Lee Y, O'Neill A (2009) Order-preserving symmetric encryption. In: Joux A (ed). Advances in Cryptology - EUROCRYPT 2009. Lecture Notes in Computer Science, vol. 5479. Springer, Berlin Heidelberg. pp 224–241
15. Kamara S, Papamanthou C, Roeder T (2012) Dynamic searchable symmetric encryption. In: Proceedings of the 2012 ACM Conference on Computer and Communications Security. CCS '12. ACM, New York, NY, USA. pp 965–976
16. Song DX, Wagner D, Perrig A (2000) Practical techniques for searches on encrypted data. In: Proceedings of the 2000 IEEE Symposium on Security and Privacy (S&P). IEEE Computer Society, Washington, DC, USA. pp 44–55
17. Chang Y-C, Mitzenmacher M (2005) Privacy preserving keyword searches on remote encrypted data. In: Ioannidis J, Keromytis A, Yung M (eds).

Applied Cryptography and Network Security. Lecture Notes in Computer Science, vol. 3531. Springer, Berlin Heidelberg. pp 442–455
18. Bellare M, Boldyreva A, O'Neill A (2007) Deterministic and efficiently searchable encryption. In: Menezes A (ed). Advances in Cryptology - CRYPTO 2007. Lecture Notes in Computer Science, vol. 4622. Springer, Berlin Heidelberg. pp 535–552
19. Kerschbaum F, Vayssière J (2008) Privacy-preserving data analytics as an outsourced service. In: Proceedings of the 2008 ACM Workshop on Secure Web Services. SWS '08. ACM, New York, NY, USA. pp 87–96
20. Li J, Omiecinski ER (2005) Efficiency and security trade-off in supporting range queries on encrypted databases. In: Proceedings of the 19th Annual IFIP WG 11.3 Working Conference on Data and Applications Security. DBSec'05. Springer, Berlin, Heidelberg. pp 69–83
21. Islam M, Kuzu M, Kantarcioglu M (2012) Access pattern disclosure on searchable encryption: Ramification, attack and mitigation. In: Proceedings of the Network and Distributed System Security Symposium (NDSS). Internet Society, Geneva, Switzerland
22. Damiani E, De Capitani di Vimercati S, Jajodia S, Paraboschi S, Samarati P (2003) Balancing confidentiality and efficiency in untrusted relational DBMSs. In: Proceedings of the ACM Conference on Computer and Communications Security (CCS). ACM, New York, NY, USA
23. Pang H, Zhang J, Mouratidis K (2013) Enhancing access privacy of range retrievals over (B^+)-trees. IEEE Transactions on Knowledge and Data Engineering 25(7):1533–1547
24. Capitani di Vimercati S, Foresti S, Paraboschi S, Pelosi G, Samarati P (2013) Distributed shuffling for preserving access confidentiality. In: Crampton J, Jajodia S, Mayes K (eds). Computer Security – ESORICS 2013. Lecture Notes in Computer Science, vol. 8134. Springer, Berlin Heidelberg. pp 628–645
25. Foresti S (2011) Preserving Privacy in Data Outsourcing. Springer, Berlin Heidelberg
26. Aggarwal G, Bawa M, Ganesan P, Garcia-Molina H, Kenthapadi K, Motwani R, Srivastava U, Thomas D, Xu Y (2005) Two can keep a secret: A distributed architecture for secure database services. In: Proceedings of the Second Conference on Innovative Data Systems Research. CIDR, Asilomar, CA, USA
27. Henrich C, Huber M, Kempka C, Mueller-Quade J, Reussner R (2010) Technical report: Secure cloud computing through a separation of duties. Institut für Kryptographie und Sicherheit (KIT)
28. Ganapathy V, Thomas D, Feder T, Garcia-Molina H, Motwani R (2011) Distributing data for secure database services. In: Proceedings of the 4th International Workshop on Privacy and Anonymity in the Information Society. PAIS '11. ACM, New York, NY, USA. pp 8–1810
29. Paillier P (1999) Public-key cryptosystems based on composite degree residuosity classes. In: Stern J (ed). Advances in Cryptology – EUROCRYPT '99. Lecture Notes in Computer Science, vol. 1592. Springer, Berlin Heidelberg. pp 223–238
30. Okamoto T, Uchiyama S (1998) A new public-key cryptosystem as secure as factoring. In: Nyberg K (ed). Advances in Cryptology – EUROCRYPT'98. Lecture Notes in Computer Science, vol. 1403. Springer, Berlin Heidelberg. pp 308–318
31. Rivest RL, Adleman L, Dertouzos ML (1978) On data banks and privacy homomorphisms. Foundations of Secure Computation 4(11):169–180
32. Hacigümüş H, Iyer B, Mehrotra S (2004) Efficient execution of aggregation queries over encrypted relational databases. In: Lee Y, Li J, Whang K-Y., Lee D (eds). Database Systems for Advanced Applications. Lecture Notes in Computer Science, vol. 2973. Springer, Berlin Heidelberg. pp 125–136
33. Mykletun E, Tsudik G (2006) Aggregation queries in the database-as-a-service model. Data Appl Security XX 4127:89–103
34. Goldreich O, Ostrovsky R (1996) Software protection and simulation on oblivious rams. Journal of the ACM 43(3):431–473
35. Stefanov E, van Dijk M, Shi E, Fletcher C, Ren L, Yu X, Devadas S (2013) Path oram: An extremely simple oblivious ram protocol. In: Proceedings of the 2013 ACM SIGSAC Conference on Computer and Communications Security. CCS '13. ACM, New York, NY, USA. pp 299–310
36. Stefanov E, Shi E (2013) Multi-cloud oblivious storage. In: Proceedings of the 2013 ACM SIGSAC Conference on Computer and Communications Security. CCS '13. ACM, New York, NY, USA. pp 247–258
37. Chor B, Kushilevitz E, Goldreich O, Sudan M (1998) Private information retrieval. Journal of the ACM 45(6):965–981
38. Olumofin F, Goldberg I (2010) Privacy-preserving queries over relational databases. In: Proceedings of the 10th International Conference on Privacy Enhancing Technologies. PETS'10. Springer, Berlin, Heidelberg. pp 75–92
39. Aguilar-Melchor C, Gaborit P (2007) A lattice-based computationally-efficient private information retrieval protocol. In: Proceedings of the Western European Workshop on Research in Cryptology (WEWoRC'2007), Bochum, Germany. pp 50–54
40. Kushilevitz E, Ostrovsky R (1997) Replication is not needed: single database, computationally-private information retrieval. In: Proceedings of the 38th Annual Symposium on Foundations of Computer Science, 1997. IEEE Computer Society, Washington, DC, USA. pp 364–373
41. Chor B, Gilboa N (1997) Computationally private information retrieval (extended abstract). In: Proceedings of the Twenty-ninth Annual ACM Symposium on Theory of Computing. STOC '97. ACM, New York, NY, USA. pp 304–313
42. Sion R, Carbunar B (2007) On the computational practicality of private information retrieval. In: Proceedings of the Network and Distributed Systems Security Symposium. Internet Society, Geneva, Switzerland
43. Beimel A, Stahl Y (2003) Robust information-theoretic private information retrieval. In: Security in Communication Networks. Springer, Berlin Heidelberg. pp 326–341
44. Beimel A, Ishai Y, Kushilevitz E, Raymond J-F (2002) Breaking the o(n1(2k-1)/) barrier for information-theoretic private information retrieval. In: Proceedings of the 43rd Annual IEEE Symposium on Foundations of Computer Science, 2002. IEEE Computer Society, Washington, DC, USA. pp 261–270
45. Jünemann K, Köhler J, Hartenstein H (2012) Data outsourcing simplified: Generating data connectors from confidentiality and access policies. In: Proceedings of the 2012 12th IEEE/ACM International Symposium OnCluster, Cloud and Grid Computing (CCGrid). pp 923–930