
Christian Henrich, Matthias Huber, Carmen Kempka, Jörn Müller-Quade, and Ralf Reussner

†IKS/EISS ‡IPD
Fakultät für Informatik
Karlsruhe Institute of Technology (KIT)
name.surname@kit.edu

This paper was submitted to the European Symposium on Research in Computer Security (ESORICS 2010)

Abstract. Cloud Computing offers many opportunities but also introduces new risks. A user who outsources a database into the Cloud loses control over his data. While the data can be secured against external threats using standard techniques, the service providers themselves have to be trusted to ensure privacy. This work proposes a novel approach to provide security for database services without the need to trust the provider. We suggest employing a separation of duties by distributing critical information and services among multiple providers in a way that the secrecy of a database can only be compromised if corrupted providers work together. We present a formal security model to evaluate distribution procedures and apply it to our approach. A result of independent interest shows that our novel security property implies k-anonymity for deterministic anonymization procedures.

1 Introduction

Due to advances in networking and virtualization technology, new paradigms of providing IT infrastructure, computing and software have emerged – among them the so-called Cloud Computing. The National Institute of Standards and Technology [20] defines Cloud Computing as “a model for enabling convenient, on-demand network access to a shared pool of computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction.”

Inherent to this model are privacy problems [21, 14]. By using services in the Cloud, clients lose control over their data. Current security mechanisms focus on protecting the data transfer to and from the service provider. But the threat of insider attacks keeps many potential customers from using Cloud Computing in critical or sensitive scenarios (e.g., scenarios comprising business secrets or customer data).

For a pure storage service, providing protection against insider attacks can be easily achieved by encrypting all data on the client side. As this prevents the server from per-
forming any meaningful operation on the data, more complex services require advanced techniques.

There are cryptographic methods like secure multiparty computation [11, 6] or private information retrieval [9] that in principle can solve all privacy problems, especially since a fully homomorphic encryption method [10] was discovered in 2009 which allows calculations on encrypted data. However, due to high communication and computation costs, these methods are infeasible and their costs outweigh all benefits of outsourcing.

Nevertheless, we need privacy and security guarantees for Cloud Computing in order to use it in sensitive scenarios. In this paper, we propose a new security notion that can be applied to outsourced databases as well as an architecture for services. We suggest partitioning a service on the basis of its duties and deploying the parts on different servers. For two examples we show that it is possible to provably provide a certain level of privacy using Separation of Duties, assuming the adversary has access to only one server. In contrast to secret sharing, this approach respects algorithms and data structures and thus preserves the efficiency of the services.

This paper is organized as follows: In the remainder of this section, we give an overview about anonymity properties and discuss related work. In Section 2, we demonstrate a subliminal channel in anonymization processes that produce $k$-anonymous databases and introduce our new security notion for anonymization procedures. We present our proposed architecture and deployment in Section 3 giving two examples and proving that they fulfill our security notion. Section 4 summarizes our results and states open problems.

1.1 Anonymity Properties

In contrast to releasing data in statistical form (macrodata), the release of specific data (microdata) allows for more detailed analysis but raises privacy concerns. To protect privacy and provide some anonymity guarantees, statistical notions have been defined that released databases have to fulfill, namely $k$-anonymity [22, 7], $l$-diversity [19], and $t$-closeness [18]. Since our security notion we define in Section 2.2 is based on $k$-anonymity the remainder of this section is focused on this property.

The idea of $k$-anonymity is that for each entry in a database that can be mapped to an individual, there are at least $k - 1$ other entries that can also be mapped to the same individual:

**Definition 1.** $k$-anonymity (taken from [7])

Let $T(A_1, ..., A_m)$ be a table, and $QI$ be a quasi-identifier associated with it, where a quasi-identifier is a set of attributes included in the private table, also externally available and therefore exploitable for linking. $T$ is said to satisfy $k$-anonymity with respect to $QI$ iff each sequence of values in $T[QI]$ appears at least with $k$ occurrences in $T[QI]$. $T[QI]$ denotes the projection, maintaining duplicate tuples, of attributes $QI$ in $T$.

In order to achieve $k$-anonymity one can generalize attribute values. The highest generalization of an attribute is called suppression. Consider for example the tables in Figure 1(a) and Figure 1(b). The table in Figure 1(b) is a 4-anonymous version of the
table in Figure 1(a), where the rows with ids 1–4 and 9–12 each form a 4-bucket. The attributes Zip Code, Age and Nationality are quasi-identifiers, while the attribute Condition is sensitive information that must not be linked to an individual. While the values of the attribute Age were generalized to achieve $k$-anonymity, the values of the attribute Nationality were suppressed.

<table>
<thead>
<tr>
<th>id</th>
<th>Zip Code</th>
<th>Age</th>
<th>Nationality</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>13053</td>
<td>28</td>
<td>Russian</td>
<td>Heart Dis.</td>
</tr>
<tr>
<td>2</td>
<td>13068</td>
<td>29</td>
<td>American</td>
<td>Heart Dis.</td>
</tr>
<tr>
<td>3</td>
<td>13068</td>
<td>21</td>
<td>Japanese</td>
<td>Viral Inf.</td>
</tr>
<tr>
<td>4</td>
<td>13053</td>
<td>23</td>
<td>American</td>
<td>Viral Inf.</td>
</tr>
</tbody>
</table>

(id)

<table>
<thead>
<tr>
<th>id</th>
<th>Zip Code</th>
<th>Age</th>
<th>Nationality</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>130**</td>
<td>&lt;30</td>
<td>*</td>
<td>Cancer</td>
</tr>
<tr>
<td>2</td>
<td>130**</td>
<td>&lt;30</td>
<td>*</td>
<td>Cancer</td>
</tr>
<tr>
<td>3</td>
<td>130**</td>
<td>&lt;30</td>
<td>*</td>
<td>Cancer</td>
</tr>
<tr>
<td>4</td>
<td>130**</td>
<td>&lt;30</td>
<td>*</td>
<td>Cancer</td>
</tr>
</tbody>
</table>

(b)

Fig. 1. (a) An example of a medical database (from [19]) with the quasi-identifiers Zip Code, Age and Nationality and the information column Condition and (b) a 4-anonymous version of this database. The attributes Zip Code and Age were generalized and Nationality was suppressed. The entries with ids 1–4 and 9–12 each form a 4-bucket.

It is well known that $k$-anonymity has a weakness [19, 18]. Consider for example the records with id 9,10,11 and 12 in Figure 1(b). If you know that there is someone in the database who is older than 29 and lives in a city with a zip code beginning with 130, you can infer that he/she has cancer (homogeneity attack). A similar attack can be done on the records with id 1,2,3 and 4 if you can rule out that an individual has a heart disease due to background knowledge. These attacks are possible because of a lack of diversity in the values of the sensitive attribute in a $k$-bucket.

Another weakness of $k$-anonymity is based on the fact that this property describes the resulting database and not the anonymization process itself. We will discuss this weakness in Section 2.1

1.2 Related Work

There are cryptographic solutions for two or more parties cooperatively computing a certain function over a set of data without any party learning anything about the input of other parties except what is learned by the output. Using an interactive protocol, these secure multiparty computations [11, 6] can thus solve all privacy problems. The problem is that for each party, the computation cost is higher than computing the whole function on the complete input without any other party. This makes the concept of multiparty computation for outsourcing services unsuitable and in fact pointless if the client is the only one with private input.
The problem of a searchable encryption is motivated by the database community having realized the benefits of outsourcing databases early. In 2002, the concept of Database as a Service was introduced by Hacigümüş et al. in [13]. For data privacy, they propose encryption and evaluate several different ways to encrypt data in a database. However, the user has to hand the encryption key to the server for query processing. This is a security risk in an untrusted server scenario. In [12], Hacigümüş et al. propose a tuple level encryption scheme and coarse indices to enable the server to execute SQL on the encrypted data. The server does not need to decrypt the data for coarse grained query execution, and returns a superset of the tuples queried. The client has to decrypt the returned data and execute the exact query on it. There were other papers considering different aspects of this idea. Damiani et al. considered confidentiality in encrypted databases in [8]. They introduce exposure coefficients and examine databases where an index refers to just one attribute value. However, in contrast to our approach, they do not provide any security notions for their schemes. A more detailed view on exposure coefficients which considers coarse indices is given by Ceselli et al. in [5]. In [15] Hore et al. addressed the problem of creating an optimal bucketization scheme under efficiency aspects.

In contrast to the schemes described above, Aggarwal et al. propose in [1] to split a database according to privacy constraints and to distribute it to different providers. They propose to use encryption if a privacy constraint cannot be met and present three encryption schemes, namely one-time pad, deterministic encryption and random addition. They also propose adding noise to enhance privacy of a distributed database. They define a composition as privacy preserving if all privacy constraints are met. However, this definition depends on the privacy constraints of the actual database. It is unclear what level of privacy can be provided if the associations between deterministically encrypted attribute values are not hidden.

Kantarcioglu and Clifton showed in [16] that classical cryptographic definitions do not work for encrypted databases because generally it is infeasible to realize a database that complies with these definitions. Therefore, they come up with new definitions and propose an encrypted database with hardware support. In [2] Amanatidis et al. look at searchable encryption from a provable-security methodology point of view. They propose an encryption scheme capable of efficient exact match and range queries while providing provable security. In [4], Bellare et al. propose a new security notion called PRIV. In contrast to our notion, a PRIV-secure database provides provable privacy only if the plaintext is drawn from a space of high min-entropy. They emphasize that the proposed schemes enable sublinear search complexity which is important for databases to enable fast query processing.

1.3 Our Contribution

Our contribution is twofold. On the one hand, we introduce a new security notion that can be applied to outsourced databases. On the other hand, we introduce our concept of Separation of Duties that allows us to separate and distribute services. Distributing services is not entirely new [1], however, to our knowledge, we are the first to give a formalized security notion for it. And, in contrast to classical secret sharing, our concept respects algorithms and data structures. This application specific distribution maintains
the efficiency of services. We apply Separation of Duties to two examples, and show that these examples fulfill our new notion. Although our CRM database example was designed with access logs in mind, we only consider static analysis, since our notion does not cover access history.

2 Security of Anonymization Functions

Though $k$-anonymity is a good first indication for anonymity, it cannot directly be used as a notion of security of services. Consider a storage service where the database is anonymized by probabilistically encrypting each entry. The resulting database clearly is not $k$-anonymous, as each combination of ciphertexts will probably be unique. However, encrypting a database this way should be considered secure by any reasonable notion of security.

This shows that a good anonymization may not be $k$-anonymous. Even worse, an anonymization process may contain a subliminal channel leaking information without violating the $k$-anonymity of the resulting database.

2.1 Subliminal Channel in $k$-Anonymity

For a database $d$ there are in general different anonymized versions complying with $k$-anonymity. Let $D^k_d$ be the set of all $k$-anonymous versions of a database $d$. A malicious database anonymizer can encode data simply by disclosing a particular database $d' \in D^k_d$. Of course the anonymizer cannot choose an arbitrary $d' \in D^k_d$.

<table>
<thead>
<tr>
<th>age</th>
<th>salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>123</td>
</tr>
<tr>
<td>31</td>
<td>234</td>
</tr>
<tr>
<td>35</td>
<td>123</td>
</tr>
<tr>
<td>35</td>
<td>234</td>
</tr>
</tbody>
</table>

(a) | 3* 1**
| 3* 2**
| 3* 1**
| 3* 2**

(b)

Fig. 2. An example salary database $d$ with two attributes (a) and the 2-anonymous version $d'$ of $d$ (b), where the attribute age was generalized once and the attribute salary twice.

Since $k$-anonymity is defined with respect to quasi-identifiers, we do not consider other columns here. Our example database $d$ (cf. Figure 2(a)) consists of two attributes. Assuming that the attribute age has two (e.g. $31 \rightarrow 3* \rightarrow *$) generalizations and the attribute salary has three (e.g. $123 \rightarrow 12* \rightarrow 1** \rightarrow *$) generalizations, we can build a generalization graph for $d$.

This graph is depicted in Figure 3. Let $d'$ (cf. Figure 2(b)) be the database the anonymizer discloses. Note that the anonymizer knows all the databases in this graph, but the adversary only learns the disclosed database $d'$. However, if the adversary knows the underlying anonymization procedure, what we assume according to Kerckhoffs’
principle, she can reconstruct the structure of the graph. Of course she neither learns if any of the databases next to or above \(d'\) are \(k\)-anonymous nor the databases themselves.

However it is still possible to introduce a subliminal channel into the anonymization procedure. One possibility is that the anonymizer scans the graph for a level of nodes containing only databases complying with the \(k\)-anonymity property, and chooses one of them. Since an adversary can reconstruct the structure of the whole graph, she also knows how many databases are on the same level as the released one. This results in a subliminal channel with capacity \(\log(n_1)\), where \(n_1\) is the number of databases on that level. Another possibility is that the anonymizer ignores inner nodes of the graph and chooses one resulting path towards the completely suppressed database \(d^*\). He releases the first database complying with the \(k\)-anonymity property he encounters. The adversary can also reconstruct which path the anonymizer took, resulting in a subliminal channel with capacity \(\log(n_2)\), where \(n_2\) is the number of disjoint paths in the graph. However, this only works if every path has a database complying with the \(k\)-anonymity property besides the trivial one \(d^*\) (cf. Figure 3).

While this may seem infeasible for an actual attack, these two examples for subliminal channels in the anonymity property \(k\)-anonymity show that we need to define anonymity not for the database that results from anonymization, but for the process of anonymization itself. We will prove that for deterministic anonymization functions which operate on the entries of a database our new notion of security implies \(k\)-anonymity. Given a process of anonymization which we consider to be ideal, we will derive a notion of security of services in Section 2.3.

2.2 Indistinguishability under Independent Column Permutation

Our security notion we describe in this chapter is motivated by \(k\)-anonymity. Intuitively, the aim of \(k\)-anonymity is to hide each individual in a crowd of size at least \(k\). Our no-
tion shares this property while overcoming the weaknesses of $k$-anonymity mentioned above. It allows for probabilistic encryption and at the same time prevents any subliminal channel that leaks information about associations between database entries. In order to achieve this, instead of examining only the anonymized table, our security notion regards the anonymization process as a whole.

More formally, we consider an anonymization function $f$ that takes a database of a certain format as input and computes an anonymized database. In the following, we differentiate between columns of quasi-identifiers and columns with sensitive information which we call information columns.

Without loss of generality, we consider databases with exactly one information column as multiple information columns can be combined. We define anonymity considering the worst case where each entry in the information column is unique. In the case of non-unique or absent information items we require that our notion holds even if absent information items are added and all information items are renamed to be unique. For our purposes, we define a database as follows:

**Definition 2.** A database is a table containing arbitrary many quasi-identifier columns and exactly one information column. All entries in the information column are unique within this column. We call the set of all databases $DB$.

As argued above, this captures databases in full generality.

**Definition 3.** A database function is a function $f : DB \rightarrow DB$. We call $D$ the set of all database functions.

Examples for database functions are projection $\pi$ and selection $\sigma$ from relational algebra and our anonymization functions. Another special case of database functions we will use are special permutations defined as follows:

**Definition 4.** The set of column independent permutations $\Pi \subset D$ is the set of database functions $p : DB \rightarrow DB$ such that each $p \in \Pi$ permutes entries within each quasi-identifier column of a database but leaves the information column untouched. The permutations implied by each $p \in \Pi$ on the individual columns can be independent from each other.

Please note that the identity function is a valid permutation. Therefore a permutation according to the definition above does not have to affect all quasi-identifier columns.

Intuitively, we say $f$ is a good anonymization if an adversary is unable to distinguish between an anonymization of the original database and an anonymization of a version of the original database where the entries in each column have been permuted independently from each other and hence relations between entries have been eliminated. This intuitive notion cannot always be met, and in the definition below, we will allow to restrict the permutations within the columns to affect only entries in $k$ rows. This restriction to $k$ rows is motivated from $k$-anonymity. To define this formally, we use a technique often used in cryptography: In an experiment an adversary $A$ has to distinguish whether she has got the result of an anonymization of a database $d$ or the anonymization of a permutation $p(d)$.

Let $d \in DB$ be a database, $f \in D$ a database function, $p \in \Pi$ a column independent permutation, $A$ an adversary and $i \in \{0, 1\}$. Depending on $i$ we define our experiment as follows:
\[ \text{Ind-ICP}_{ad}^{k,p} \]
\[ d_0 := f(d) \]
\[ d_1 := f(p(d)) \]
\[ b := \mathcal{A}(d_i) \]
\[ \text{return } b \]

In the experiment \( \text{Ind-ICP}_{ad}^{k,p,0} \), \( \mathcal{A} \)'s input is \( f(d) \), an anonymization of the original database \( d \). In the experiment \( \text{Ind-ICP}_{ad}^{k,p,1} \), \( \mathcal{A} \) gets \( f(p(d)) \), an anonymization of \( p(d) \) which is a permutation of the original database, as input.

Using this experiment we define the notion \( k \)-Indistinguishability under Independent Column Permutation with \( \varepsilon \)-Advantage.

**Definition 5.** (\( k \)-Ind-ICP with \( \varepsilon \)-Advantage)

For a database function \( f \) \( k \)-Indistinguishability under Independent Column Permutation (\( k \)-Ind-ICP) with \( \varepsilon \)-Advantage holds iff for each polynomially restricted adversary \( \mathcal{A} \), for each database \( d \in \text{DB} \) the following holds:

For each row \( r_i \) in \( d \) exists a set \( M_i \) of \( k \) rows in \( d \) (called the \( k \)-bucket of row \( r_i \)) such that \( r_i \in M_i \) and for each \( p \in \Pi \) that affects only rows in \( M_i \) and leaves the entries of all other rows unchanged the advantage of the adversary \( \mathcal{A} \) is

\[
\text{Adv}_{ad}^{\text{Ind-ICP}}(d) := \left| \Pr[\text{Ind-ICP}_{ad}^{k,p,0}(d) = 1] - \Pr[\text{Ind-ICP}_{ad}^{k,p,1}(d) = 1] \right| < \varepsilon.
\]

If \( k \)-Ind-ICP holds for an anonymization function \( f \) and a reasonably small \( \varepsilon \), a subliminal channel in \( f \) will at most leak information about individual entries in the original database. It does not contain information about relations between entries that would be eliminated by independently permuting the entries of each column. Therefore, relations between any quasi-identifiers and sensitive attributes are hidden by \( f \).

In the rest of this paper we consider \( \varepsilon \) to be reasonably small. In some cases, e.g., when encryption is involved, it makes sence to define \( \varepsilon \) depending on a security parameter \( s \) which could for example be the key length of the used encryption. More precisely, for a family of anonymization functions and a security parameter \( s \) one could demand that \( \varepsilon \) decreases asymptotically in \( s \).

**Theorem 1.** If \( k \)-Ind-ICP with \( \varepsilon \)-Advantage holds for a reasonably small \( \varepsilon \) for an anonymization function \( f \) that operates deterministically on the entries within each \( k \)-bucket \( M_i \), then \( f(d) \) is \( k \)-anonymous for all \( d \in \text{DB} \) which have either 0 or at least \( k \) rows.

Note that in particular a fully deterministic function operating on the entries satisfies the requirement of deterministic behavior on each \( k \)-bucket \( M_i \).

**Proof.** The proof is by contradiction. Let \( f \) be a database function which is \( k \)-Ind-ICP with \( \varepsilon \)-Advantage and fulfills the preconditions of Theorem 1. Further let \( d \in \text{DB} \) be a database for which \( f(d) \) is not \( k \)-anonymous. Then there exists a row \( r_j \) in the database \( f(d) \) with a combination of identifiers which occurs less than \( k \) times in the database \( f(d) \). For the row corresponding to \( r_j \) in the original database \( d \) there exists a \( k \)-bucket \( M_j \) of \( k \) rows, such that the image of \( f \) is indistinguishable if the entries of each column are permutated within \( M_j \).
Let $M'$ denote an arbitrary set having this property and let $f(M')$ denote the rows in $f(d)$ corresponding to $M'$. There must be one row $r_i \in f(M')$ for which at least one quasi-identifier differs from the corresponding quasi-identifier in row $r_j$. As $f$ operates deterministically on the quasi-identifiers within $M'$, we have that exchanging the quasi-identifiers from the rows corresponding to $r_i$ and $r_j$ will exchange the quasi-identifiers from the rows $r_i$ and $r_j$ and hence this permutation does not yield indistinguishable images under the mapping $f$. As this is true for arbitrary $M'$, the function $f$ cannot have been $k$-Ind-ICP with $\epsilon$-advantage for any reasonably small $\epsilon$.

One gets an even stronger notion if one adds $l$-diversity to $k$-Ind-ICP by requiring the images of $f$ to be $l$-diverse.

**Definition 6.** Let $f$ be a database function for which $k$-Ind-ICP with $\epsilon$-advantage holds. The function $f$ is called $l$-Div-$k$-Ind-ICP with $\epsilon$-advantage if for each $d \in DB$ with at least $k$ rows $f(d)$ is $l$-diverse.

For the sake of readability in the rest of this paper, we write “$k$-Ind-ICP” instead of “$k$-Ind-ICP with $\epsilon$-advantage for a reasonably small $\epsilon$”.

### 2.3 Ideal Anonymization

Given a class of anonymization procedures $F$ which are assumed to be ideal, we define the security of a real anonymization process by comparison. We say a real process $g$ anonymizes a class $DB$ of databases as good as $F$ if for $g$ there exists a function $f \in F$ such that for all $d \in DB$ intuitively the following holds: Everything that can be learned from $g(d)$ which was anonymized using the real process can also be learned from the ideally anonymized $f(d)$. More formally we say that for each polynomial time adversary $A$ which is given $g(d)$ there exists a polynomial time simulator $S$ which is given $f(d)$ such that the outputs of $A$ and $S$ are computationally indistinguishable.

With this concept giving a concrete definition of security boils down to specifying the class $F$ which specifies the amount of anonymity that is to be reached.

A concrete class $F$ of ideal anonymization procedures could be the class of all $k$-Ind-ICP mappings with $\epsilon$-advantage for certain $k$ and $\epsilon$. A more strict choice of the class $F$ would be to first apply a $k$-Ind-ICP mapping and then encrypting each entry with an ideal cipher. However, the latter class of ideal anonymization procedures can be too strict. Sometimes facts can be derived from an anonymized database which are commonly known, but would formally violate the security definition. To cope with this the ideal anonymization functions from $F$ may also output some auxiliary information which will explicitly not be kept secret. Examples for this common knowledge are that the height of persons lies within the range of zero to three meters, that a certain street is in a certain city, or that the database contains citizens of a certain region.

The amount of auxiliary information given is hence made explicit in our security notion and this amount is also a good measure for comparing different anonymization processes.

From this notion of anonymization we derive a notion of security for the Separation of Duties approach. The real anonymization process $g$ for the Separation of Duties approach is defined by the assumptions restricting the adversary. As an example let
two servers $S_1$ and $S_2$ be given and let $f_1$ and $f_2$ denote the anonymization for $S_1$ and $S_2$ respectively. For an adversary limited to compromise only one of the two servers the Separation of Duties is at least as good as an anonymization $\mathcal{F}$ if both $f_1$ and $f_2$ anonymize as good as an ideal anonymization $\mathcal{F}$.

3 Separation of Duties

To reduce the information a single provider can learn, we propose to combine the design pattern partitioned application [17] with the need to know principle [23] and apply it to services. This means separating a service with respect to its algorithms and data structures and deploying each part on a different server. Each part only gets the data needed to fulfill its duty. Simple assumptions about the servers’ capabilities (like limited or no storage) result in provable security guarantees if the separation procedure follows certain rules and an adversary only has access to one server. On a sidenote, since the code of the service is partitioned and distributed as well, this also impedes software piracy.

The remainder of this section is structured as follows. In Section 3.1 we present a general description of an architecture of a separated service and in Section 3.2 we give instructive examples to explain how our approach can enhance privacy. The examples were chosen to exhibit very simple proofs showing the easy applicability of the methods presented in this work. We show that the mapping of the database to each single server in our examples are $k$-Ind-ICP. Hence, under the assumption that the adversary can break into at most one server, our Separation of Duties approach is at least as secure as the class $\mathcal{F}$ of ideal anonymizations, where $\mathcal{F}$ denotes the class of $k$-Ind-ICP mappings with $k$ being the number of rows in the database.

3.1 Description of the Proposed Architecture

Separating a service into different parts can be done in two ways which we term serial and parallel. When a service is separated serially into two sub-services or components, one part calls the other part (cf. Figure 4(a)). Consider for example a database storage service calling an indexing service for faster search. With serial separation, in order to be connected the components or the servers the components are deployed on have to be aware of each other. With parallel separation, this is not necessary. However, for such a separation to be transparent to the client application, an adaptor has to be deployed. Figure 4(b) shows a parallel separation with an adaptor component deployed on the client’s machine. The adaptor or the client application respectively has to call both parts. For example consider the sequence diagram in Figure 5. Here a client calls the adaptor. The adaptor calls the indexing service and then queries the database using the results from the indexing service. With parallel separation one can achieve potentially more secure services than with serial separation since the components do not need to be aware of each other. This reduces the likelihood of malicious collaboration. However, parallel separation of services may be harder to achieve since it may require a new component as well as an adaption of existing components and the calling party. This may be counterintuitive for software engineers. A more detailed view on how to separate services and to deploy the parts on different servers is given in Section 3.2.
Fig. 4. System and deployment view of a (a) serial and (b) parallel separation of a service. The composition structure and the deployment relationship cross each other. This is not in conflict with UML and conceptionally sound, as long as the inner components are bundled by an assembly where inner components remain visible, exactly for the reason of possibly distributed deployment. Both concepts, composed components, which would hide inner structural details, and assemblies are not well separated in the UML, but in more modern component models, such as Palladio[3].

3.2 Examples

In this section we give two instructive examples in order to explain how Separation of Duties enhances the security of services. We use our new security notion defined in Section 2.2 to make explicit statements about their security. We provide proofs that these separations provide privacy according to our notion.

CRM Database Consider the example customer relationship database in Figure 6. We treat all attributes as quasi-identifiers and want to hide any connection between them. In this CRM database we want to search for a name and get the corresponding bank account number, but will never search for a bank account number. For this example, we propose to separate the indices from the database, and apply encryption (cf. Figures 7 and 8). We use a deterministic encryption $\text{ENC}_{\text{det}}$ and a probabilistic encryption $\text{ENC}_{\text{prob}}$, and introduce random numbers $r_i$ used as pointers to connect the distributed entries. Keywords are encrypted with $\text{ENC}_{\text{det}}$ deterministically to enable searching, the pointers and the attribute values are encrypted using $\text{ENC}_{\text{prob}}$ to prevent the server to gain information about correlations.

Consider the system depicted in Figure 9. In this example, the tables in Figures 7 and 8 are each on a different server. The name index component stores the table of Figure 7(a), the surname index component the table of Figure 7(b), respectively. The associations database component stores the table of Figure 8. Because of this separation, before retrieving or writing data to the associations database, the client has to
query the index component(s) for pointers or create new pointers respectively. For example if the client wants to execute the query “SELECT * FROM table WHERE name = Alice AND surname = Smith;”, the client has to query the name index with subquery1 = “SELECT * FROM table WHERE keyword = ENCdet(Alice);”, the surname with subquery2 = “SELECT * FROM table WHERE keyword = ENCdet(Smith);” and finally query the associations database server with the decrypted results.

Note that, despite the communication and encryption overhead, search time is still sublinear. Also note that this example is not secure in the strict sense of classical cryptography as information about the structure of the data, e.g., the number of different attribute values, may leak.

**Security Proof** Let $d$ be a database of the form displayed in Figure 6. We first consider the indexing servers and show that the function $f_1$ mapping $d$ to indexing databases of the form displayed in Figure 7 as described above is $k$-Ind-ICP. The function $f_1$ sup-

---

**Fig. 5.** Sequence diagram for a query on an example of a parallelly separated database service

**Fig. 6.** An example of a CRM database.

---

**Fig. 7.** Separated indices of the database with the deterministically encrypted names and the probabilistically encrypted pointers.
presses all attributes except a given attribute \( a \), deletes duplicate values of that attribute and encrypts the remaining values deterministically with \( ENC_{\text{det}} \).

In order to obtain the pointers for the association, \( f_1 \) adds a column containing a unique random number for each occurrence of the corresponding value of \( a \) in the original database. Then \( f_1 \) encrypts all entries in this column probabilistically with a given encryption algorithm \( ENC_{\text{prob}} \). The result of \( f_1 \) generates the indexing tables described above.

The database \( f_1(d) \) contains two columns. In the first column, no two values are identical since we removed duplicates before applying the deterministic encryption. The entries in the second column are random values, so (assuming a good encryption) the table is indistinguishable from a table with mere random entries of the same size. Hence \( f_1 \) is \( k \)-Ind-ICP, because permutations of the entries of each column cannot be detected after an application of \( f_1 \).

Second we look at the association server. Let \( f_2 \) be a function that replaces every attribute value of a given database with a unique random number and additionally \( f_2 \) adds a column containing original attribute values of each row encrypted with a given secure encryption algorithm \( ENC_{\text{prob}} \). The mapping \( f_2 \) generates exactly the associations table described above. We will show that \( f_2 \) is \( k \)-Ind-ICP for \( k \) being the number of rows of the database \( d \). Since \( f_2(d) \) only contains unique random numbers or plaintext encrypted with \( ENC_{\text{prob}} \), we have that permutations of the entries of each column cannot be detected after an application of \( f_2 \) and \( f_2 \) clearly is \( k \)-Ind-ICP.

**Intelligent Electric Meter** Consider an intelligent decentralized power grid, where the provider keeps track of the power consumption of its clients to dynamically activate or deactivate power stations or consumers. Intelligent electric meters in the clients’ system report the consumption to a central database. Clients, however, do not want the provider to know their current consumption for privacy reasons. The requirements of both the
provider and the consumers can be met by separating and distributing the storage service. Consider the system depicted in Figure 10. Here, the service is serially separated into two components. If clients send their consumption encrypted with the consumption database’s public key the anonymizer does not learn their current consumption, but still can anonymize the data. Note that this example is not secure in the strict sense of classical cryptography as information about the data, e.g., names or consumption rates, may leak as only the correlation is hidden.

Figure 11 depicts an example database the anonymizer may hold. Clients send their consumption encrypted to the anonymizer, which adds for every tuple “client - ENC(consumption)” a unique random number. The anonymizer sends the relation “ENC(consumption) - random number” to the consumption database, which can decrypt the values of the consumption attribute with his private key. An example for the

<table>
<thead>
<tr>
<th>client</th>
<th>ENC(consumption)</th>
<th>random number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Henry Smith</td>
<td>xXzzs43Hfd</td>
<td>573535635</td>
</tr>
<tr>
<td>John Doe</td>
<td>5DF5dbctl</td>
<td>436343346</td>
</tr>
</tbody>
</table>

Fig. 11. The association table of the anonymizer component mapping clients to random number, effectively anonymizing each client’s consumption information.

Fig. 12. The consumption information as seen by the consumption database component.

Security Proof To show that our Separation of Duties approach in this example is at least as secure as the class of $k$-Ind-ICP mappings, we have to consider the anonymizer and the server containing the consumption database. The first server learns the original
user identities and the encrypted power consumption. As encrypted values of power
consumptions are indistinguishable, a permutation of the entries of each column before
encrypting the power consumption entries cannot be detected and the mapping of the
original database to the anonymizer’s data is $k$-Ind-ICP. The second server learns the
original power consumption entries of each row, but obtains the rows in permuted or-
der with randomized user IDs. A permutation of the entries within the columns before
introducing the randomized IDs and applying the permutation of the rows cannot be
detected and so the second mapping is $k$-Ind-ICP, too.

4 Conclusion and Future Work

Cloud Computing introduces serious privacy risks to which classical cryptographic
techniques cannot be applied as they undo the advantages of outsourcing services. In
this work, we introduced an approach to distribute data and services to several indepen-
dent servers while respecting the algorithms and data structures used in the service. Our
Separation of Duties approach does not offer the same level of security as a complete
encryption of a database, but allows for services to be performed efficiently on the data.

The abstract definitions from classical cryptography cannot capture application spe-
cific security and more fine grained notions are necessary. In order to give precise secu-
ri ty guarantees, we introduced a novel security notion. This notion allows to precisely
judge even weak levels of protection which cannot be expressed in the security mod-
els of classical cryptography. We showed that $k$-anonymity is inapplicable for judging
anonymization mechanisms as it disregards subliminal channels in an anonymization
procedure and is unfitting for encrypted attribute values. Our notion, inspired by $k$-
anonymity, describes the anonymization procedure and not the results and so can detect
and prevent subliminal channels. This notion hides relations within a $k$-bucket and is
applicable to practical anonymization procedures generating databases which are not
$k$-anonymous but intuitively secure. We gave two practical examples and suggested an
anonymization process for each. The protection mechanisms given provably meet our
new security notion.

For future work, we propose to consider dynamic attacks where an adversary learns
access statistics. Also, an examination of subliminal channels of anonymization func-
tions enforcing $l$-diversity is necessary. This could lead to stronger notions of anonymity
and to notions which can measure the effectiveness of introducing dummy data or
dummy queries to further enhance the security of Cloud Computing.

References

Kenthapadi, Rajeev Motwani, Utkarsh Srivastava, Dilyys Thomas, and Ying Xu. Two can
keep a secret: A distributed architecture for secure database services. CIDR 2005.
2. Georgios Amanatidis, Alexandra Boldyreva, and Adam O’Neill. Provably-secure schemes
3. Steffen Becker, Heiko Koziol ek, and Ralf Reussner. The Palladio component model for
18. Ninghui Li and Tiancheng Li. t-closeness: Privacy beyond k-anonymity and l-diversity. 2007.