Closeness Constraints for Separation of Duties in Cloud Databases as an Optimization Problem

Ferdinand Bollwein[1] and Lena Wiese[2]

[1] Institute of Computer Science, TU Clausthal
ferdinand.bollwein@tu-clausthal.de
[2] Institute of Computer Science, University of Goettingen
wiese@cs.uni-goettingen.de

Abstract. Cloud databases offer flexible off-premise data storage and data processing. Security requirements might however impede the use of cloud databases if sensitive or business-critical data are accumulated at a single cloud storage provider. Hence, partitioning the data into less sensitive fragments that are distributed among multiple non-communicating cloud storage providers is a viable method to enforce confidentiality constraints. In this paper, we express this enforcement as an integer linear program. At the same time visibility of certain data combinations can be enabled. Yet in case of violated visibility constraints, the number of different servers on which data is distributed can still be optimized. We introduce novel closeness constraints to express these requirements.

1 Introduction

Cloud databases are a generic tool for outsourcing not only data storage but also data processing: cloud databases offer advanced query and manipulation languages to create database schemas, insert data into tables, query data based on some conditions, update and delete data. Moreover, cloud databases offer joins and aggregation functions. Hence a typical business application of cloud databases is that a cloud customer uploads data into the cloud database and locally only runs scripts to retrieve and manage data on the customer side. This relieves the cloud customer from the burden to install, configure and update a large-scale database system on customer side. Furthermore, depending on changing customer needs, the storage capacity can flexibly be reduced or expanded. However, when private and business-critical data are processed by the cloud database as unencrypted plaintext, cloud database customers have to put a high level of trust in a confidentiality-preserving and privacy-compliant treatment of the data. One way to reduce this trust is to enable the cloud customer to manage distribution of data on as many providers (and under as many user names) as necessary to avoid harmful accumulation of data at a single site.

Separation of duties for cloud databases means that data are split into fragments and these fragments are stored on independent cloud providers. In this paper, vertical fragmentation is used as a technique to protect data confidentiality in cloud databases. Consistently with related work the confidentiality

requirements are modeled as subsets of attributes of the relations. The resulting fragments are explicitly linkable however, it is assumed that they are stored on separate servers which are assumed to be non-communicating. The problem of finding such fragmentations is modeled as a mathematical optimization problem and it is one of the main objectives to minimize the number of servers involved. Moreover, constraints are introduced to improve the usability of the resulting fragmentations and to allow for efficient query answering. Those constraints are modeled as soft constraints in contrast to the confidentiality requirements which are obligatory to be satisfied.

In this paper, we make the following contributions:

- we formalize the enforcement of confidentiality constraints by obtaining *multiple* fragments as a mathematical optimization problem.
- we formalize the distribution of these fragments on *multiple* servers while at the same time minimizing the amount of these servers; that is we obtain a distribution on as few servers as possible.
- moreover, further constraints are introduced to improve the usability of the resulting fragmentations and to allow for efficient query answering; we discuss a weakness of conventional visibility constraints and introduce additional closeness constraints concerned with the distribution of the attributes to allow for efficient query processing.
- visibility and closeness constraints are modeled as soft constraints – in contrast, confidentiality constraints are hard and have to be fully satisfied.

We start this article with a survey of related work in Section 2. Section 3 sets the necessary terminology; Sections 4 and 5 analyze a standard and an extended Separation of Duties problem; Section 6 provides a translation into an integer linear program; Section 7 briefly describes a prototypical implementation; Section 8 concludes the article.

2 Related Work

Horizontal (row-wise) and vertical (column-wise) fragmentation are the two basic approaches to partition tables. Fragmentation as a security mechanism follows the assumption that links between data are highly sensitive (for example, linking a patient name with a disease) whereas individual values (only patient names or only diseases) are less sensitive. The existing approaches can be divided into:

- keep-a-few approaches: some highly sensitive data are maintained at the trusted client side while non-sensitive fragments are stored on an external server (like a cloud database); this approach was pioneered in [5].
- non-communicating servers approaches: fragments are stored on different servers that do not interact; this approach was pioneered in [1].

These approaches only consider fragmentation of a single table into *two* fragments. In the former case (keep-a-few), a server fragment and an owner fragment

is obtained; in the latter case (non-communicating servers) two fragments are obtained to be stored on two servers.

In [6] the authors consider multiple fragments however they require these fragments 1) to be unlinkable to be stored on one single external server and 2) to be non-overlapping. In contrast to this, we assume that the servers are non-communicating (and hence allow linkability in particular by tuple ID to enable recombination of results on the client side) and we allow a certain level of overlaps (and hence redundancy of data) to improve data visibility.

Vertical [2] as well as horizontal [14] confidentiality-preserving fragmentations have also been analyzed on a logical background. Last but not least, the article [10] surveys several approaches.

3 Relations and Fragmentation

In this paper we assume the common setting of a database table that has to be vertically split into fragments in order to hide some secret information. A database table consists of a set of columns (the names of which are also called attributes). Each attribute has a data type and an according domain of values denoted $\mathrm{dom}(a_i)$. More formally, we talk about a relation schema $R(A) = R(a_1, \ldots, a_n)$ that consists of a relation name R and a finite set of attributes $A = \{a_1, \ldots, a_n\}$. A *relation* (instance) r on the relation schema $R(a_1, \ldots, a_n)$, also denoted by $r(R)$, is defined as an ordered set of *n-tuples* $r = (t_1, \ldots, t_m)$ such that each *tuple* t_j is an ordered list $t_j = \langle v_1, \ldots, v_n \rangle$ of values $v_i \in \mathrm{dom}(a_i)$ or $v_i = \mathsf{NULL}$.

In order to enforce confidentiality constraints, we obtain a vertical fragmentation of the table. Each fragment contains a subset of the attributes in A. We have to define a special tuple identifier to be able to recombine the original table from the fragments. More formally, a fragmentation of a table is a set $\mathbf{f} = (f_0, \ldots, f_k)$ of fragments f_i where each f_i contains a tuple identifier *tid* (a candidate key of the relation which can itself consist of several attributes) and further attributes: $f_i = \{tid, a_{i_1}, \ldots, a_{i_k}\}$ where $a_{i_j} \in A$. The fragment f_0 is the dedicated *owner fragment* (which is in particular needed to satisfy the singleton confidentiality constraints); all other fragments $f_1, \ldots, f_m$ are *server fragments* which should be allocated on different non-communicating cloud storage providers. Following [1], due to the non-communication assumption, we allow the different server fragments to be linkable (in particular, by the tuple ID but also be common attributes to achieve higher visibility as described in a later section).

When fragmenting a relation vertically, there are two main requirements. The first one (completeness) is that every attribute must be placed in at least one fragment to prevent data loss. The second property (reconstruction) that must be satisfied is more technical: by including a candidate key in every fragment, it is possible to associate the tuples of the individual table fragments. Equijoin operations on those attributes can then be used to reconstruct the original relation. There is also a third property (disjointness) which is often required. This property demands that every non-tuple-identifier attribute is placed in exactly one vertical fragment. However, especially in the context of this work, there are

reasons to omit this property to increase the usability of the resulting vertical fragmentation. Detailed information on this is presented in Section 5. Based on this preliminary information, the *correct/lossless vertical fragmentation* of a single relation is formally defined as follows:

Definition 1 (Vertical Fragmentation). *Let r be a relation on the relation schema $R(A)$. Let $tid \subset A$, the* tuple identifier*, be a predefined candidate key of r. A sequence $\mathbf{f} = (f_0, \ldots, f_k)$ where $f_j \subseteq A$ for all $j \in \{0, \ldots, k\}$ is called a* correct vertical fragmentation *of r if the following conditions are met:*

- ***Completeness:*** $\bigcup_{j=0}^{k} f_j = A$
- ***Disjointness:*** $f_i \cap f_j \subseteq tid$, *for all* $f_i \neq f_j$ *with* $f_i, f_j \neq \emptyset$
- ***Reconstruction****:* $tid \subset f_j$, *if* $f_j \neq \emptyset$

A fragmentation that satisfies completeness and reconstruction but not necessarily the disjointness property is called a lossless vertical fragmentation *of r.*
The cardinality $\mathrm{card}(\mathbf{f})$ *of a correct/lossless vertical fragmentation of r is defined as the number of nonempty fragments of* $\mathbf{f}$*:* $\mathrm{card}(\mathbf{f}) = \sum_{\substack{j=0 \\ f_j \neq \emptyset}}^{k} 1$.
At physical level, the relation fragment *or* table fragment *derived from fragment f_j is given by the projection $\pi_{f_j}(r)$.*

It is further worth noticing that the tuple identifier is required to form a proper subset of the fragments which prohibits fragments consisting of the tuple identifier attributes only. This requirement is due to the fact that the tuple identifier's sole purpose should be to ensure the reconstruction property.

4 Standard Separation of Duties Problem

The security requirements are specified at attribute level, i.e. certain attributes or combinations of attributes are considered sensitive and must not be stored by a single untrusted database server. This can, consistently with related work [5, 1, 4, 3, 11], be modeled with the notion of *confidentiality constraints*.

A *confidentiality constraint* is a subset of attributes of a table: a confidentiality constraint is written as $c \subseteq A$. We differentiate the following two cases:

1. Singleton constraints where the cardinality $card(c) = 1$; that is, c contains only a single attribute $c = \{a_i\}$. In this case, the servers are not allowed to read the values in column a_i.
2. Association constraints (see [10]) with cardinality $card(c) > 1$. In this case, the servers are not allowed to read a combination of values of those attributes contained in the confidentiality constraints. However any real subset of these attributes may be revealed.

Definition 2 (Confidentiality Constraints). *Let $R(A)$ be a relation schema over the set of attributes A. A confidentiality constraint on $R(A)$ is defined by a subset of attributes $c \subseteq A$ with $c \neq \emptyset$. A confidentiality constraint c with $|c| = 1$ is called a singleton constraint; a confidentiality constraint c that satisfies $|c| > 1$ is called an association constraint.*

As an example, consider a table containing information about patients of a hospital. We might have highly sensitive identifying attributes like name and SSN (social security number); these would then be turned into singleton confidentiality constraints. On the other hand, some attributes are only sensitive in combination: the birth year, the ZIP code and the gender in combination can act as a quasi-identifier which can reveal a patient's identity. In this case, any subset of birth year, ZIP code and gender may be revealed but not the entire combination.

As attributes contained in a singleton constraint are not allowed to be accessed by an untrusted server, they cannot be outsourced in plaintext at all. Because we refrain from using encryption those attributes have to be stored locally at the owner side. On the other hand, association constraints can be satisfied by distributing the respective attributes among two or database servers. More precisely, a correct vertical fragmentation $\mathbf{f} = (f_0, \ldots, f_k)$ has to be found in which one fragment stores all the attributes contained in singleton constraints and all other fragments are not a superset of a confidentiality constraint. As a common convention throughout the rest of this work, fragment f_0 will always denote the *owner fragment* which stores all the attributes contained in singleton constraints. This fragment is stored by a local, trusted database. The other fragments $f_1, \ldots, f_k$ denote the *server fragments* and each of those is stored by a different untrusted database server. We require the server fragments $f_1, \ldots, f_k$ to obey a given set of confidentiality constraints $C = \{c_1, \ldots, c_l\}$. A server fragment f_j is confidentiality-preserving if $c \not\subseteq f_j$ for all $c \in C$. This leads to the formal definition of a *confidentiality-preserving vertical fragmentation*:

Definition 3 (Confidentiality-preserving Vertical Fragmentation). *For relation r on schema $R(A)$ and a set of confidentiality constraints C, a correct/lossless vertical fragmentation* $\mathbf{f} = (f_0, \ldots, f_k)$ preserves confidentiality *with respect to C if for all $c \in C$ and $1 \leq j \leq k$ it holds that $c \not\subseteq f_j$.*

It is necessary to introduce some reasonable restrictions to the set of confidentiality constraints. These restrictions are of theoretical nature and will not restrict its expressiveness. These requirements are summarized by the following definition of a *well-defined* set of confidentiality constraints (we extend the definition of e.g. [10] to our special treatment of tuple identifiers):

Definition 4 (Well-defined Set of Confidentiality Constraints). *Given a relation r on the relation schema $R(A)$ and a designated tuple identifier $tid \subset A$. A set of confidentiality constraints C is* well-defined *if it satisfies:*

- *For all $c, c' \in C$ with $c \neq c'$, it holds that $c \not\subseteq c'$.*
- *For all $c \in C$, it holds that $c \cap tid = \emptyset$.*

The first condition requires that no confidentiality constraint c is a subset of another confidentiality constraint c'. By the definition of a confidentiality-preserving vertical fragmentation, the satisfaction of c' would be redundant because $c \not\subseteq f_j$ for $j \in \{1, \ldots, k\}$ implies that $c' \not\subseteq f_j$ for $j \in \{1, \ldots, k\}$ if $c \subseteq c'$.
The second condition requires that the tuple identifier attributes are considered

insensitive on their own and in combination with other attributes. The tuple identifier's sole purpose is to ensure the reconstruction of the fragmentation by placing it in every nonempty fragment. If, for example, there would be a confidentiality constraint $c \subseteq \text{tid}$, a confidentiality-preserving vertical fragmentation would require that the corresponding tuple identifier attributes cannot be placed in any server fragment. Therefore, every attribute has to be placed in the owner fragment which basically means that the relation cannot be fragmented at all.

Storage space restrictions might also be an important factor for the vertically fragmented relation: the owner and the server fragments may not exceed a databases' capacity. Hence, we assume that there is a weight function that assigns a weight to each subset of attributes $w_r : \mathcal{P}(A) \longrightarrow \mathbb{R}_{\geq 0}$.

It is quite obvious that the cardinality of the confidentiality-preserving fragmentation to be found is a crucial factor for the quality of the fragmentation. Keeping the number of involved server as low as possible will reduce the customer's costs, lower the complexity of maintaining the vertically fragmented relation and also increase the efficiency of executing queries. Therefore, in the following problem statement, the objective is to find a confidentiality-preserving correct vertical fragmentation of minimal cardinality. Additionally, the capacities of the involved storage locations must not be exceeded. Formally, the *(Standard) Separation of Duties Problem* is hence defined as follows:

Definition 5 (Standard Separation of Duties Problem). *For relation r over schema $R(A)$, a well-defined set of confidentiality constraints C, a dedicated tuple identifier $\text{tid} \subset A$, a weight function w_r, storage spaces $S_0, \ldots, S_k$ (where S_0 denotes the owner's storage and $S_1, \ldots S_k$ denote the servers' storages) and maximum capacities $W_0, \ldots, W_k \in \mathbb{R}_{\geq 0}$. Find a correct confidentiality-preserving fragmentation $\mathbf{f} = (f_0, \ldots, f_k)$ of minimal cardinality such that the capacities of the storages are not exceeded, i.e. $w_r(f_j) \leq W_j$ for all $0 \leq j \leq k$.*

One should note that in this general formulation the owner fragment can possibly contain all of the attributes if W_0 is sufficiently large. Moreover, in order to solve the problem, one could first assign all attributes in singleton constraints to the owner fragment and afterwards solve the remaining subproblem without singleton constraints. Hence, by considering appropriate values for W_0 one can influence the size of the owner fragment and the overall resulting fragmentation.

5 Extended Separation of Duties Problem

In many scenarios, it is desirable that certain combinations of attributes are stored by a single server or in other words, these combinations are visible on a single server, because they are often queried together. This can be accounted for with the notion of *visibility constraints*:

Definition 6 (Visibility Constraint). *Let $R(A)$ denote a relation schema over the set of attributes A and let r be a relation over $R(A)$. A* visibility constraint *over $R(A)$ is a subset of attributes $v \subseteq A$. A fragmentation $\mathbf{f} =$*

$(f_0, \ldots, f_k)$ *satisfies* v *if there exists* $0 \leq j \leq k$ *such that* $v \subseteq f_j$. *In this case, define* $\mathrm{sat}_v(\mathbf{f}) \coloneqq 1$ *and* $\mathrm{sat}_v(\mathbf{f}) \coloneqq 0$ *otherwise. Furthermore, for any set* V *the number of satisfied visibility constraints is*

$$\mathrm{sat}_V(\mathbf{f}) \coloneqq \sum_{v \in V} \mathrm{sat}_v(\mathbf{f}).$$

In contrast to confidentiality constraints, the fulfillment of visibility constraints is not mandatory, i.e. confidentiality constraints are *hard constraints* while visibility constraints are *soft constraints*. Roughly speaking, the following extended version of the Separation of Duties Problem aims at finding a confidentiality-preserving vertical fragmentation that minimizes the number of fragments and maximizes the number of satisfied visibility constraints. While there is not much sense in finding a fragmentation that does not satisfy the completeness property, breaking the disjointness property can help to increase the number of satisfied visibility constraints and therefore, in the upcoming problem definition a lossless but not necessarily correct fragmentation will be required.

Although visibility constraints provide a means of keeping certain attributes close together, i.e. on a single server, they are not useful when a certain constraint cannot be satisfied due to some confidentiality constraint. Consider a relation r over the attributes $A = \{\mathsf{PatientID}, \mathsf{DoB}, \mathsf{ZIP}, \mathsf{Diagnosis}, \mathsf{Treatment}\}$ with the dedicated tuple identifier $\mathsf{PatientID}$. Moreover, let a weight function of r be defined by $w_r(a) = 1$ for all $a \in A$. Furthermore, suppose the owner fragment has a capacity of $W_0 = 0$, and there are 3 servers with capacities $W_1 = 2$, $W_2 = 3$ and $W_3 = 2$. For statistical purposes, a visibility constraint $v = \{\mathsf{DoB}, \mathsf{ZIP}, \mathsf{Diagnosis}\}$ is introduced and to preserve the privacy of the patients, the confidentiality constraint $c = \{\mathsf{DoB}, \mathsf{ZIP}\}$ is enforced. However, because $c \subset v$, the visibility constraint cannot be satisfied. Hence, one possible solution to the problem is given by $\mathbf{f} = \{f_0, f_1, f_2, f_3\}$ with: $f_0 = \emptyset$, $f_1 = \{\mathsf{PatientID}, \mathsf{DoB}\}$, $f_2 = \{\mathsf{PatientID}, \mathsf{ZIP}, \mathsf{Treatment}\}$, $f_3 = \{\mathsf{PatientID}, \mathsf{Diagnosis}\}$. Another possible solution is given by the fragmentation $\mathbf{f}' = \{f'_0, f'_1, f'_2, f'_3\}$ with: $f'_0 = \emptyset$, $f'_1 = \{\mathsf{PatientID}, \mathsf{DoB}\}$, $f'_2 = \{\mathsf{PatientID}, \mathsf{ZIP}, \mathsf{Diagnosis}\}$, $f'_3 = \{\mathsf{PatientID}, \mathsf{Treatment}\}$. The important thing to notice here is that in $\mathbf{f}$ the attributes in v are spread among three and in $\mathbf{f}'$ among only two servers. As a result, a query for the three attributes DoB, ZIP and $\mathsf{Diagnosis}$ involves three servers for the first fragmentation and only two for the second. Hence, the query will be processed faster for the second fragmentation because on the one hand, the server that stores f'_2 can evaluate conditions on both attributes ZIP and $\mathsf{Diagnosis}$ resulting in smaller intermediate results and on the other hand, there is less communication overhead due to the necessity of two servers only. Therefore, it is reasonable to provide constraints to make sure that certain attributes should be distributed among as few servers as possible. Moreover, as in the following problem statement a lossless fragmentation will be required, those constraints can also be used to limit the number of copies of any individual attribute. This introduces an interesting technique to reduce the setup time of a vertical fragmented relation. These so-called *closeness constraints* are defined as follows:

Definition 7 (Closeness Constraint). *Let $R(A)$ denote relation schema over the set of attributes A and let r be a relation over $R(A)$. A closeness constraint over $R(A)$ is a subset of attributes $\gamma \subseteq A$. Let $\mathbf{f} = (f_0, \ldots, f_k)$ be a correct/lossless vertical fragmentation of r, the distribution $\mathrm{dist}_\gamma(\mathbf{f})$ of γ is defined as the number of fragments that contain one of the attributes in γ:*

$$\mathrm{dist}_\gamma(\mathbf{f}) \coloneqq \sum_{\substack{j=0: \\ f_j \cap \gamma \neq \emptyset}}^{k} 1$$

For any set Γ of closeness constraints, the distribution $\mathrm{dist}_\Gamma(\mathbf{f})$ is defined as the sum of distributions of $\gamma \in \Gamma$.

The following extended problem definition aims at preserving confidentiality by requiring a lossless fragmentation that does not violate any confidentiality constraint. Moreover, the owner's and the servers' capacities must not be exceeded. Furthermore, the minimization of the weighted sum serves three purposes: The summand $\alpha_1 \,\mathrm{card}(\mathbf{f})$ is responsible for minimizing the cardinality of the fragmentation. By subtracting the summand $\alpha_2 \,\mathrm{sat}_V(\mathbf{f})$, each satisfied visibility constraint will lower the overall objective value. Lastly, the distribution of the closeness constraints is minimized by the summand $\alpha_3 \,\mathrm{dist}_\Gamma(\mathbf{f})$. With these explanations, the *Extended Separation of Duties Problem* is defined as follows:

Definition 8 (Extended Separation of Duties Problem). *For relation r over schema $R(A)$, a well-defined set of confidentiality constraints C, a set of visibility constraints V, a set of closeness constraints Γ, a tuple identifier $tid \subset A$, a weight function w_r, storage spaces $S_0, \ldots, S_k$, maximum capacities $W_0, \ldots, W_k \in \mathbb{R}_{\geq 0}$ and weights $\alpha_1, \alpha_2, \alpha_3 \in \mathbb{R}_{\geq 0}$. Find a lossless confidentiality-preserving fragmentation $\mathbf{f} = (f_0, \ldots, f_k)$ of minimal cardinality which satisfies $w_r(f_j) \leq W_j$ for all $0 \leq j \leq k$ such that the following weighted sum is minimized*

$$\alpha_1 \,\mathrm{card}(\mathbf{f}) - \alpha_2 \,\mathrm{sat}_V(\mathbf{f}) + \alpha_3 \,\mathrm{dist}_\Gamma(\mathbf{f}).$$

A reasonable choice for α_1, α_2 and α_3 is presented in the following. The idea is to assign priorities to the three different objectives. In most scenarios, the overall number of necessary servers will have the highest impact on the usability and therefore, minimizing it should have the highest priority. Hence, the desired solution's cardinality should be minimal. The satisfaction of visibility constraints has the second highest priority and therefore, the resulting fragmentation should minimize the cardinality of the fragmentation and the number of satisfied visibility constraints should be maximal among all other confidentiality-preserving fragmentations of minimal cardinality that do not violate the capacity constraints. Finally, among those solutions, the distribution of the closeness constraints should be minimized. This can be achieved by solving the linear inequalities $\alpha_2|V| + \alpha_3(k+1)|\Gamma| < \alpha_1$ and $\alpha_3(k+1)|\Gamma| < \alpha_2$. Solving these inequalities is straightforward and under the assumption that $|V| > 0$ and $|\Gamma| > 0$, one possible solution is given by $\alpha_1 = 1$, $\alpha_2 = \frac{0.9}{2|V|}$ and $\alpha_3 = \frac{0.87}{2(k+1)|V||\Gamma|}$.

Listing 1 Extended Separation of Duties Problem

$$\text{minimize} \quad \alpha_1 \sum_{j=0}^{k} y_j - \alpha_2 \sum_{v \in V} z_v + \alpha_3 \sum_{\gamma \in \Gamma} \sum_{j=0}^{k} \delta_{\gamma j} \tag{1}$$

$$\text{subject to} \quad \sum_{j=0}^{k} x_{ij} \geq 1, \qquad a_i \in A \tag{2}$$

$$x_{ij} = y_j, \qquad a_i \in \text{tid},\ j \in \{0, \ldots, k\} \tag{3}$$

$$\sum_{a_i \in A \setminus \text{tid}} x_{ij} \geq x_{i'j}, \qquad a_{i'} \in \text{tid},\ j \in \{0, \ldots, k\} \tag{4}$$

$$\sum_{a_i \in A} w_r(a_i) x_{ij} \leq W_j y_j, \qquad j \in \{0, \ldots, k\} \tag{5}$$

$$\sum_{a_i \in c} x_{ij} \leq |c| - 1, \qquad j \in \{1, \ldots, k\},\ c \in C \tag{6}$$

$$\sum_{a_i \in v} x_{ij} \geq u_{vj} |v|, \qquad j \in \{0, \ldots, k\},\ v \in V \tag{7}$$

$$\sum_{j=0}^{k} u_{vj} \geq z_v, \qquad v \in V \tag{8}$$

$$\sum_{a_i \in \gamma} x_{ij} \leq |\gamma| \delta_{\gamma j}, \qquad \gamma \in \Gamma,\ j \in \{0, \ldots, k\} \tag{9}$$

6 Integer Linear Program Formulation

In this section, the ILP formulation for the Extended Separation of Duties Problems as shown in Listing 1 will be discussed. All variables x_{ij}, y_j, z_v, u_{vj}, $\delta_{\gamma j}$ are binary. In order to identify which fragments should be nonempty, variables $y_0, \ldots, y_k \in \{0, 1\}$ are introduced for the owner fragment f_0 and for each server fragment $f_1, \ldots, f_k$. A value of one indicates that the respective fragment is nonempty. Furthermore, additional binary variables $x_{ij} \in \{0, 1\}$ for each $a_i \in A$ and $j \in \{0, \ldots, k\}$ are used to indicate that attribute a_i is stored in fragment f_j. Additional indicator variables $u_{vj} \in \{0, 1\}$ for all visibility constraints $v \in V$ and all fragments $j \in \{0, \ldots, k\}$ are introduced which are interpreted as follows: If $u_{vj} = 1$, all attributes in v must be stored in fragment f_j. If $u_{vj} = 0$, all attributes in v may be (but do not have to be) stored in this fragment. Moreover indicator variables $z_v \in \{0, 1\}$ are used to indicate that visibility constraint v is satisfied by at least one fragment. This means that z_v can be equal to one if at least one u_{vj} equals one. Moreover, additional variables $\delta_{\gamma j} \in \{0, 1\}$ for all closeness constraints $\gamma \in \Gamma$ and every fragment $j \in \{0, \ldots, k\}$ are necessary to express that fragment f_j contains one or more attributes of γ.
The objective function (1) minimizes the weighted sum stated in Definition 8 in terms of the variables y_j, z_v and $\delta_{\gamma j}$. Because the Extended Separation of Duties Problem only requires a lossless fragmentation, there is no condition that ensures

the disjointness property. Constraint (2) ensures the completeness property by requiring that for each $a_i \in A$ there exists at least one j such that x_{ij} equals one. The following Constraint (3) requires that if a fragment is nonempty, it must include the tuple identifier because if $y_j = 1$ all x_{ij} for all $a_i \in \text{tid}$ must be equal to one. Conversely, if the fragment should be empty, i.e. $y_j = 0$, no tuple identifier attribute should be placed in the fragment and therefore, x_{ij} must be equal to zero for each $a_i \in \text{tid}$. In the definition of fragmentation, the tuple identifier is required to be a proper subset of each non-empty fragment. This is achieved by Constraint (4) because every tuple identifier attribute $a_{i'} \in \text{tid}$ can only be placed in a fragment f_j, i.e $x_{i'j} = 1$, if there is at least one non-tuple-identifier attribute a_i placed in the same fragment, i.e. $x_{ij} = 1$. Condition (5) has two functions. On the one hand, if fragment f_j should be nonempty and $y_j = 1$, it ensures that the servers capacity W_j is not exceeded. On the other hand, if $y_j = 0$ and f_j should be empty, all x_{ij} for $a_i \in A$ must equal zero and therefore, no attribute can be stored in that fragment. Side constraint (6) makes sure that at most $\text{card}(c) - 1$ attributes contained in a confidentiality constraint are stored in the same server fragment f_j for $j \in \{1, \ldots, k\}$. On the one hand, this ensures that all attributes in a singleton constraint are stored in the owner fragment and on the other hand that no association constraint is violated. Conditions for the visibility constraints are (7) and (8). Each z_v for all $v \in V$ lowers the objective value if $z_v = 1$. Constraint (8) allows $z_v = 1$ only if one of the u_{vj} is is equal to one. However, due to condition (7), a variable u_{ij} can only take a value of one if $x_{ij} = 1$ for all $a_i \in v$. This means that visibility constraint v is satisfied by fragment f_j. Constraint (9) ensures that for each closeness constraint γ and each fragment f_j, the variable $\delta_{\gamma j}$ can only be zero if no attribute $a_i \in \gamma$ is stored in fragment f_j. Therefore, the distribution of γ and the objective value increases for every fragment f_j that contains an attribute in γ.
From an ILP solution, the fragments f_j can be derived by building the sets:

$$f_j \coloneqq \begin{cases} \{a_i \in A \mid x_{ij} = 1\}, & \text{if } y_j = 1 \\ \emptyset, & \text{else} \end{cases}$$

These fragments then form a correct vertical fragmentation as required in the problem statement.

It should be mentioned further that in some scenarios some visibility or closeness constraints might be more important to satisfy than others. If this is the case, one can simply introduce weights $\beta_v \in (0, 1]$ for all visibility constraints $v \in V$ and weights $\beta_\gamma \in (0, 1]$ for all $\gamma \in \Gamma$ and use the objective function

$$\alpha_1 \sum_{j=0}^{k} y_j - \alpha_2 \sum_{v \in V} \beta_v z_v + \alpha_3 \sum_{\gamma \in \Gamma} \beta_\gamma \sum_{j=0}^{k} q_{\gamma j}$$

in the ILP formulation. This way, visibility constraints with higher weight will contribute more to the minimization of the objective function. Moreover, reducing the distribution of closeness constraints with higher weights is more important than reducing the distribution of closeness constraints with smaller weights.

7 Prototype and Evaluation

We implemented a prototype fragmentation and distribution system (available at http://www.uni-goettingen.de/de/558180.html) based on the IBM ILOG CPLEX solver and PostgreSQL. For testing we set up a TCP-H benchmark (http://www.tpc.org/tpch/) on a single PC equipped with an Intel Xeon E3-1231v3 @3.40GHz (4 Cores), 32GB DDR3 RAM and a Seagate ST2000DM001 2TB HDD with 7200 rpm running Ubuntu 16.04 LTS. The database servers ran in separate, identical virtual machines which are assigned 4 cores and 8GB of RAM. The virtual machines are running Ubuntu Server 16.04 LTS with an instance of PostgreSQL 9.6.1 installed. We implemented the distributed setting using foreign data wrapper extension postgres_fdw. On the trusted server hosting the owner fragment we created views for the remote server fragments. We ran all 22 queries of the TPC-H benchmark against a non-fragmented local and against the fragmented installation. It turned out that Postgres was not able to process queries Q_{20} and Q_{17} not even in the unfragmented case and we stopped execution after 30 minutes. Apart from these, for the view-based queries Table 1 shows the execution time (t) in seconds and the slow down (sd) compared to the execution time of the same query on the original database (ot).

Q	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	18	19	21	22
t	41.18	4.699	19.8	18.6	37.0	4.039	11.65	38.58	18.53	12.5	0.58	10.4	11.8	3.765	8.69	2.98	51.0	1.93	79.111	10.58
ot	2.267	0.353	0.861	3.11	0.95	0.291	0.530	1.305	1.652	1.4	0.19	0.457	1.7	0.341	0.66	0.6	5.99	0.65	1708.5	0.534
sd	18.16	13.31	22.99	5.97	38.9	13.88	21.99	29.57	11.22	8.85	2.98	22.75	6.85	11.04	13.1	4.95	8.51	2.98	0.05	19.81

Table 1. TPC-H queries (seconds) on fragments (t), unfragmented (ot), slowdown (sd)

Overall, the increase in execution time compared to the queries on the non-fragmented database does not follow a specific pattern. The slowdown on the distributed views was always less than 30 times – one query even executed faster on the distributed installation. Execution time hence very much depends on the query plan PostgreSQL establishes. To fully understand what causes the increase in the execution times, one would have to study the execution strategy for each of the queries individually; one could then develop strategies to achieve better performances for queries on the vertically fragmented database.

8 Discussion and Conclusion

We studied the problem of finding a confidentiality-preserving vertical fragmentation as a mathematical optimization problem. To achieve a better distribution of attributes among the servers we introduced closeness constraints in addition to conventional visibility constraints.

In future work, we plan to combine the presented approach with partial encryption of a table similar to several approaches surveyed in [10]. Balancing the amount of encrypted and non-encrypted columns leaves room for further mathematical optimization problems. Moreover combining fragmentation with existing

frameworks using novel property-preserving encryption schemes (like in [7–9, 12, 13]) offers even more options to balance leakage and distribution. Because sensitive associations cannot only occur between columns but also between rows of a database, another interesting extension of this work is to additionally explore horizontal fragmentation (as in [14]) which means that database tables are fragmented and distributed row-wise.

References

1. Aggarwal, G., Bawa, M., Ganesan, P., Garcia-Molina, H., Kenthapadi, K., Motwani, R., Srivastava, U., Thomas, D., Xu, Y.: Two can keep a secret: A distributed architecture for secure database services. In: The Second Biennial Conference on Innovative Data Systems Research (CIDR 2005) (2005)
2. Biskup, J., Preuß, M., Wiese, L.: On the inference-proofness of database fragmentation satisfying confidentiality constraints. In: ISC. Lecture Notes in Computer Science, vol. 7001, pp. 246–261. Springer (2011)
3. Ciriani, V., Di Vimercati, S.D.C., Foresti, S., Jajodia, S., Paraboschi, S., Samarati, P.: Fragmentation and encryption to enforce privacy in data storage. In: European Symposium on Research in Computer Security. pp. 171–186. Springer (2007)
4. Ciriani, V., De Capitani di Vimercati, S., Foresti, S., Jajodia, S., Paraboschi, S., Samarati, P.: Selective data outsourcing for enforcing privacy. Journal of Computer Security 19(3), 531–566 (2011)
5. Ciriani, V., di Vimercati, S.D.C., Foresti, S., Jajodia, S., Paraboschi, S., Samarati, P.: Keep a few: Outsourcing data while maintaining confidentiality. In: ESORICS. Lecture Notes in Computer Science, vol. 5789, pp. 440–455. Springer (2009)
6. Ciriani, V., Vimercati, S.D.C.D., Foresti, S., Jajodia, S., Paraboschi, S., Samarati, P.: Combining fragmentation and encryption to protect privacy in data storage. ACM Transactions on Information and System Security (TISSEC) 13(3), 22 (2010)
7. Popa, R.A., Redfield, C., Zeldovich, N., Balakrishnan, H.: Cryptdb: protecting confidentiality with encrypted query processing. In: Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles. pp. 85–100. ACM (2011)
8. Sarfraz, M.I., Nabeel, M., Cao, J., Bertino, E.: Dbmask: fine-grained access control on encrypted relational databases. In: Proceedings of the 5th ACM Conference on Data and Application Security and Privacy. pp. 1–11. ACM (2015)
9. Spillner, J., Beck, M., Schill, A., Bohnert, T.M.: Stealth databases: Ensuring user-controlled queries in untrusted cloud environments. In: 8th International Conference on Utility and Cloud Computing. pp. 261–270. IEEE (2015)
10. di Vimercati, S.D.C., Erbacher, R.F., Foresti, S., Jajodia, S., Livraga, G., Samarati, P.: Encryption and fragmentation for data confidentiality in the cloud. In: Foundations of Security Analysis and Design VII, pp. 212–243. Springer (2014)
11. di Vimercati, S.D.C., Foresti, S., Jajodia, S., Livraga, G., Paraboschi, S., Samarati, P.: Fragmentation in presence of data dependencies. IEEE Transactions on Dependable and Secure Computing 11(6), 510–523 (2014)
12. Waage, T., Homann, D., Wiese, L.: Practical application of order-preserving encryption in wide column stores. In: SECRYPT. pp. 352–359. SciTePress (2016)
13. Waage, T., Jhajj, R.S., Wiese, L.: Searchable encryption in apache cassandra. In: Foundations and Practice of Security. pp. 286–293. Springer (2015)
14. Wiese, L.: Horizontal fragmentation for data outsourcing with formula-based confidentiality constraints. In: IWSEC. Lecture Notes in Computer Science, vol. 6434, pp. 101–116. Springer (2010)