A Universal Approach to Guarantee Data Privacy

Thomas Studer

Abstract. The problem of data privacy is to verify that confidential information stored in an information system is not provided to unauthorized users and, therefore, personal and other sensitive data remain private. One way to guarantee this is to distort a knowledge base such that it does not reveal sensitive information. In the present paper we will give a universal definition of the problem of knowledge base distortion. It is universal in the sense that is independent of any particular knowledge representation formalism. We will then present a basic and general algorithm for knowledge base distortion to guarantee data privacy. This algorithm provides us with upper bounds for the complexity of the distortion problem. Moreover, we examine heuristics to improve its average performance.

Mathematics Subject Classification (2010). 03B70; 68T27.

Keywords. Data privacy, controlled query evaluation, inference control, lying, knowledge base systems, propositional logic, description logic.

1. Introduction

Our modern information society is based on the sharing and integration of information. Almost all organizations (public and private) need to collect and automatically process lots of data where some of it, in particular personally identifiable information, needs protection. That means access to that information should be restricted such that data privacy can be guaranteed. Although technology alone cannot address all privacy concerns, it is important that information systems take responsibility for the privacy of the data they manage, see for instance [1].

Information stored in a system usually is protected against unauthorized access such that users only are allowed to access a limited portion of it. In this situation the following questions arise:

Research supported by the SNSF grant 200021-132070.

1. What can a user infer from the information to which he has access?
2. Can the system guarantee that a user cannot obtain knowledge about sensitive information?

Controlled query evaluation [3, 4, 5] is a very successful approach to privacy preserving query answering. The idea is that each time a query is evaluated it is checked whether the answer would leak sensitive information to the user. If this is the case, then the answer to the query is distorted. There are basically two possible distortion methods: the answer can simply be refused [15] or the system can give an incorrect answer (that is it lies) [8]. The framework of controlled query evaluation has been applied to a variety of data models and control mechanisms, see for instance [3, 4, 5]. However, most of these applications have been made in the area of complete systems. A remarkable exception is [6], which, like the present paper, deals with incomplete logical databases.

Recently, a static variant of controlled query evaluation has been developed [7]. There the idea is to construct, in a preprocessing step, a so-called *inference proof* or *privacy preserving* database or knowledge base that can respond to any query without leaking private information. Of course, this inference proof database should be as close as possible to the original unprotected database. That means as little changes as necessary have been made to it and hence the system only lies when it is absolutely necessary for not leaking sensitive information.

In this paper, we will present a first general study of this static approach to controlled query evaluation from the perspective of knowledge base systems. We will give the fundamental algorithms and establish basic results about them. We start with the following general definition which will be made precise in Section 2.

The *problem of knowledge base distortion* for data privacy consists in finding a subset $\mathcal{KB}'$ of a given knowledge base $\mathcal{KB}$ such that $\mathcal{KB}'$ is a maximal privacy preserving subset of $\mathcal{KB}$.

This definition is universal in the sense that it does not depend on a particular knowledge representation language. Also in Section 2 we will introduce our running example which is used to illustrate all notions and algorithms.

We will then in Section 3 introduce a universal algorithm to find such a maximal privacy preserving subset for any (unsecure) knowledge base. The algorithm implements a simple depth first search strategy which, again, does not depend on the knowledge representation language. It only assumes that we can decide whether a knowledge base preserves privacy. Therefore, the algorithm cannot make use of sophisticated techniques and is more or less straightforward, which has the advantage that it can be applied to a great variety of formalisms.

In Section 4 we will make use of this algorithm to establish upper bounds on the complexity of the knowledge base distortion problem. Using an oracle, we give a general theorem that again does not depend on the formalism used

for knowledge representation. However, we will also state corollaries about the complexities of the distortion problem for particular languages and logics such as classical propositional logic and several description logics.

The algorithm as presented before performs an uninformed search. Depending on the procedure that decides whether a knowledge base is privacy preserving, we obtain heuristics that improve the performance of the search algorithm. The basic idea is that in case a knowledge base $\mathcal{KB}$ is not privacy preserving, the decision procedure cannot only return the value false but it can provide information about which elements of $\mathcal{KB}$ are responsible for the security breach. We can then use this information to remove those elements from $\mathcal{KB}$ first that leak the most information, that is those elements that are used in the most derivations of secrets. This is closely related to the method of axiom pinpointing for description logics [14] that originally has been developed to provide explanations why a certain consequence holds in a knowledge base. We will present in Section 5 a variant of our algorithm that implements such a heuristic and thus operates in an informed way. Again, we will use our running example to support this heuristic and to illustrate its impact.

Finally Section 6 concludes the paper and hints at future research directions in the realm of knowledge base distortion to guarantee data privacy.

2. Problem setting

In this section we set the formal stage for the paper. We give definitions of the notions we will employ and we present the precise statement of the problem we are going to examine. Moreover, we introduce the running example to illustrate our approach.

We consider languages simply to be sets of formulas (without further structure). In particular, we need two languages $\mathcal{L}_A$ and $\mathcal{L}_S$ (note that they may be equal) where

1. $\mathcal{L}_A$ is used to state the assertions stored in the knowledge base and
2. $\mathcal{L}_S$ is used to specify the secrets that should not be revealed to the public.

Thus we define:

1. a *knowledge base* $\mathcal{KB}$ is a finite set of formulas of $\mathcal{L}_A$,
2. a *secret* S is a formula of $\mathcal{L}_S$.

We will use $\mathcal{SC}$ to denote finite sets of secrets.

We also need a decidable and monotone *consequence relation* Con between knowledge bases and secrets. Thus we set

$$\mathsf{Con} \subseteq \mathfrak{P}_{\mathsf{fin}}(\mathcal{L}_A) \times \mathcal{L}_S,$$

where $\mathfrak{P}_{\mathsf{fin}}$ denotes the finite power set operator. If a secret S is a consequence of a knowledge base $\mathcal{KB}$, formally $\mathsf{Con}(\mathcal{KB}, S)$, then this means that the secret can be inferred from the knowledge base. In other words, if an agent has access to the knowledge base, then using the information from the knowledge base

he may derive the secret S. In that case, the secret is revealed and privacy is violated.

Definition 1. A knowledge base $\mathcal{KB}$ *preserves privacy* with respect to a set of secrets $\mathcal{SC}$ if for each $S \in \mathcal{SC}$ we have $\neg\mathsf{Con}(\mathcal{KB}, S)$.

Usually, a user also has a priori knowledge, which is knowledge the user has without accessing the knowledge base. This may be, for instance, general background knowledge about the application domain. In our model such a priori knowledge is integrated in the consequence relation. The following example shows how the consequence relation takes care of the a priori knowledge

Example 2. Our running example is concerned with medical information which, of course, is highly sensitive. Hospitals in Switzerland are legally obligated to report detailed information about the administered treatments to the Swiss Federal Statistical Office (SFSO). The SFSO uses this information to annually create statistics about health care in Switzerland [10]. The collected data includes sociodemographic information about the patients (sex, age, region of residence, etc.) as well as administrative data (kind of insurance, length of stay in hospital, diagnosis). Here we will not use real world data from this statistics (see [16] for privacy issues of the real world data), but only use the same kind of information the SFSO statistics provide.

We assume that the knowledge base $\mathcal{KB}$ includes the following facts.

1. If a patient lives in Region A, then that patient's diagnosis is *broken leg*, *cancer*, or *aids*:
$$\mathsf{RegionA} \rightarrow \mathsf{brokenLeg} \vee \mathsf{cancer} \vee \mathsf{aids}. \tag{A.1}$$
2. Patient 1 lives in Region A:
$$\mathsf{Patient1} \rightarrow \mathsf{RegionA}. \tag{A.2}$$
3. Patient 1 does not receive a high cost treatment:
$$\neg(\mathsf{Patient1} \rightarrow \mathsf{highCosts}). \tag{A.3}$$

Further, we assume that the following facts are a priori knowledge:

1. A cancer diagnosis entails a high cost treatment:
$$\mathsf{cancer} \rightarrow \mathsf{highCosts}. \tag{P.1}$$
2. An aids diagnosis entails a high cost treatment:
$$\mathsf{aids} \rightarrow \mathsf{highCosts}. \tag{P.2}$$

For a knowledge base T and a secret S, we define the consequence relation Con by

$$\mathsf{Con}(T, S) \quad \text{if and only if} \quad T \cup \{(\text{P.1}), (\text{P.2})\} \models S$$

where $\models$ is the entailment relation of classical propositional logic. Note that (P.1) and (P.2) need not be part of T but still may contribute to the reasoning process.

For our knowledge base $\mathcal{KB}$, we find, for example, that

$$\mathsf{Con}(\mathcal{KB}, \mathsf{Patient1} \rightarrow \mathsf{brokenLeg}).$$

That means if an agent has access to the knowledge base, then he may infer that Patient 1 has the diagnosis *broken leg*. Hence, privacy is violated if we assume that a patient's diagnosis is sensible information that should be kept secret. Again note that the knowledge base alone would not allow for the conclusion $\mathsf{Patient1} \rightarrow \mathsf{brokenLeg}$. It is the knowledge base plus the additional a priori knowledge that leads to the information leakage.

In this example both $\mathcal{L}_A$ and $\mathcal{L}_S$ are the language of propositional logic and we also use this language to represent the a priori knowledge and the definition of Con. However, that need not necessarily be the case. For example, in a description logic based approach, the a priori knowledge may be ontological knowledge which is expressed via subsumption relations between concepts (expressed in a TBox) whereas the knowledge base may be assertions (in the form of an ABox). The consequence relation Con would then be defined as entailment with respect to the TBox.

We are now in the position to state the general definition of the knowledge base distortion problem.

Definition 3. The *distortion problem* for a knowledge base $\mathcal{KB}$ with respect to a set of secrets $\mathcal{SC}$ is to find a subset $\mathcal{KB}'$ of $\mathcal{KB}$ such that

1. $\mathcal{KB}'$ preserves privacy with respect to $\mathcal{SC}$ and
2. for all $\mathcal{KB}'' \subset \mathcal{KB}$ with $\mathsf{card}(\mathcal{KB}') < \mathsf{card}(\mathcal{KB}'')$ we have that $\mathcal{KB}''$ does not preserve privacy with respect to $\mathcal{SC}$.

We call such a knowledge base $\mathcal{KB}'$ a *solution* of the distortion problem.

A solution $\mathcal{KB}'$ of the distortion problem for $\mathcal{KB}$ with respect to a set of secrets $\mathcal{SC}$ is a maximal subset of $\mathcal{KB}$ that preserves privacy for $\mathcal{SC}$. Note that such a solution need not be unique. In Example 2, for instance, we could remove any of the three facts from $\mathcal{KB}$ and we would obtain a knowledge base that does preserve privacy with respect to $\mathcal{SC} = \{\mathsf{Patient1} \rightarrow \mathsf{brokenLeg}\}$.

According to our definition, solutions are maximal with respect to cardinality. Another possibility would be to require a solution to be maximal with respect to the subset relation. Clause 2 of Definition 3 would then read:

for all $\mathcal{KB}'' \subset \mathcal{KB}$ with $\mathcal{KB}' \subsetneq \mathcal{KB}''$ we have that $\mathcal{KB}''$ does not preserve privacy with respect to $\mathcal{SC}$.

However, this leads to different solutions than Definition 3 as the following example shows.

Example 4. Let $\mathcal{KB}$ consist of (A.1), (A.2), (A.3) plus the following:

1. Patient 2 lives in Region A:
$$\mathsf{Patient2} \rightarrow \mathsf{RegionA}. \tag{A.4}$$
2. It is not the case that Patient 2's diagnosis is *broken leg*:
$$\neg(\mathsf{Patient2} \rightarrow \mathsf{brokenLeg}). \tag{A.5}$$

We want to hide two secrets:

$$\mathcal{SC} = \{\mathsf{Patient1} \rightarrow \mathsf{brokenLeg}, \mathsf{Patient2} \rightarrow \mathsf{highCosts}\}$$

where the second secret means that Patient 2 receives a high cost treatment.

Further, let Con be the same as in Example 2. As above, $\mathcal{KB}$ is not privacy preserving with respect to $\mathcal{SC}$ and the question is what elements do we have to remove from $\mathcal{KB}$ to obtain a solution to the distortion problem.

Simple logical reasoning shows that $\mathcal{KB}' := \mathcal{KB} \setminus \{(\text{A.1})\}$ preserves privacy with respect to $\mathcal{SC}$. Indeed, $\mathcal{KB}'$ is a solution to the distortion problem since it is a maximal privacy preserving knowledge base. The only knowledge base with greater cardinality is $\mathcal{KB}$ which does not preserve privacy.

The knowledge base $\mathcal{KB}'' := \mathcal{KB} \setminus \{(\text{A.2}), (\text{A.4})\}$ also preserves privacy. However, it is not a solution to the distortion problem since there is a knowledge base with greater cardinality (namely $\mathcal{KB}'$) that preserves privacy. However, $\mathcal{KB}''$ is a solution with respect to subsets since none of its strict supersets preserves privacy with respect to $\mathcal{SC}$. Note that it is easy to find a solution with respect to subsets: one can simply remove element by element from the knowledge base until privacy is preserved. Finding a solution with respect to cardinality is harder. It the next section we present a backtracking algorithm that performs a bounded search to find such a solution.

3. Algorithms

We are going to present a depth first search algorithm to solve the distortion problem for a knowledge base $\mathcal{KB}$. The basic idea of a depth first search is as follows. Check whether $\mathcal{KB}$ is privacy preserving. If so, answer $\mathcal{KB}$; otherwise remove an element from $\mathcal{KB}$ which gives a knowledge base $\mathcal{KB}'$ and check whether that is privacy preserving. If so, answer $\mathcal{KB}'$; otherwise remove an element from $\mathcal{KB}'$ which gives a knowledge base $\mathcal{KB}''$ and check whether that is privacy preserving and so on.

Of course, this algorithm only finds a solution that is maximal with respect to the subset relation. In order to obtain a solution that fits our definition (that means it is maximal with respect to cardinality), the algorithm cannot immediately return a privacy preserving knowledge base but it has to backtrack and check whether it finds a larger one. Only if it has tried all knowledge bases, it can be sure to have found the largest one. However, once the algorithm has found a privacy preserving knowledge base, there is a trivial way to prune the search space: the algorithm only has to consider knowledge bases that have greater cardinality than the one already found.

In the following we provide a pseudo code implementation for such a depth first search procedure. It consists of the following three functions where the argument A is a knowledge base.

1. `depthFirst`(A) is the main function. It is a simple wrapper function that calls `doBoundedSearch`$(A, -1, \emptyset)$.

2. `doBoundedSearch`$(A, bound, closed)$ recursively performs a depth first search for a solution to the distortion problem. The search tree is pruned if the knowledge base that is currently investigated does not have more than *bound* elements. Moreover it only looks for solutions that contain *closed* as a subset.
3. `isPP`(A) tests whether A is privacy preserving.

We assume that the set of secrets $\mathcal{SC}$ is globally available and hence we do not pass it as an argument to `isPP`. Let us now give some comments on the implementation of `doBoundedSearch`. First, it is checked, whether A is privacy preserving. If so, A is returned; otherwise subsets of A are searched as follows where the variable *result* stores the best solution found so far and *bound* stores its size. An element a of A which does not belong to *closed* is removed from A which gives a knowledge base C. If the size of C is greater than *bound*, then `doBoundedSearch` is called recursively with the arguments C, *bound*, and *closed*. There it is important that *closed* is passed as value and not as reference (or that one passes a reference to a copy of *closed*). That is to guarantee that the call to `doBoundedSearch` does not change the value of *closed*. When this recursive call returns, all subsets of $A \setminus \{a\}$ have been searched and if a privacy preserving subset has been found, the variables *result* and *bound* are updated. The assertion a is then added to *closed*, which means that in the following only subsets of A that contain a will be searched. This process is iterated for all a in A that do not belong to *closed*.

We saw in Example 2 that the solution to the distortion problem need not be unique. Our algorithm, as we presented it, will only compute one solution but not produce the complete list of all solutions. However, it is easy to adapt it such that it will answer all solutions. Instead of looking for privacy preserving subsets with size greater than the current bound the algorithm should look for possible solutions with size greater *or equal* than the current bound. If such a knowledge base has been found then

1. its is added to the list of solutions if its size equals the bound,
2. the list of solutions is cleared (all its elements are dropped) and the newly found solution is added (this is then the only element of the list) if its size is greater than the current bound.

Another possible adaption concerns the problem that our algorithm treats all axioms of the knowledge bases in the same way. It does not take into account that some axioms may be more important than others and that it should first try to remove the less important axioms. It may even be better to remove two less important axioms than one important axiom. This problem can be solved by assigning a weight to each axiom in the knowledge base such that the more important an axiom is the greater its weight is. The search algorithm is then adapted such that it no longer looks for a knowledge base with maximal cardinality but for one with maximal weight. That means the sum of the weights of its axioms should to be maximal.

Algorithm 1 `depthFirst`(A)

Require: A is a knowledge base
Ensure: returns a solution to the distortion problem for A with respect to a given set of secrets $\mathcal{SC}$
Ensure: returns null if no subset of A preserves privacy with respect to $\mathcal{SC}$

return `doBoundedSearch`$(A, -1, \emptyset)$

Algorithm 2 `doBoundedSearch`$(A, bound, closed)$

Require: A is a knowledge base with $\mathsf{card}(A) > bound$
Require: $bound$ is an integer
Require: $closed$ is a subset of A
Ensure: returns a solution to the distortion problem for A with respect to a given set of secrets $\mathcal{SC}$ which contains more than $bound$ elements and which is a superset of $closed$
Ensure: returns null if no such solution exists

if `isPP`(A) **then**
  return A
end if
$result \leftarrow \mathsf{null}$
$B \leftarrow A \setminus closed$
for all $a \in B$ **do**
  $C \leftarrow A \setminus \{a\}$
  if $\mathsf{card}(C) > bound$ **then**
    $subResult \leftarrow$ `doBoundedSearch`$(C, bound, closed)$
    if $subResult \neq \mathsf{null}$ **then**
      $bound \leftarrow \mathsf{card}(subResult)$
      $result \leftarrow subResult$
    end if
  end if
  $closed \leftarrow closed \cup \{a\}$
end for
return $result$

Example 5. We will illustrate the algorithm with an example run. Note that we choose not the best order in the forall loop. This is to show the pruning of the search tree and to later compare this algorithm with the informed search.

Consider the knowledge base $\mathcal{KB}$ and the set of secrets $\mathcal{SC}$ from Example 4. An example run of the uninformed search makes calls to `isPP` with the following arguments:

1. $\mathcal{KB}$.
2. $\mathcal{KB} \setminus \{(\text{A.2})\}$.
3. $\mathcal{KB} \setminus \{(\text{A.2}), (\text{A.3})\}$.

Algorithm 3 `isPP`(A)

Require: A is a knowledge base
Ensure: returns true if A preserves privacy with respect to $\mathcal{SC}$
Ensure: returns false otherwise

```
privacyPreserving ← true
i ← 1
while privacyPreserving and i ≤ card(SC) do
  privacyPreserving ← not(Con(A, SC_i))
  i ← i + 1
end while
return privacyPreserving
```

4. $\mathcal{KB} \setminus \{(\text{A.2}), (\text{A.3}), (\text{A.4})\}$. This is a privacy preserving knowledge base with size 2. In the following only knowledge bases with size larger than 2 will be considered.
5. $\mathcal{KB} \setminus \{(\text{A.2}), (\text{A.4})\}$. This is a privacy preserving knowledge base with size 3. In the following only knowledge bases with size larger than 3 will be considered.
6. $\mathcal{KB} \setminus \{(\text{A.3})\}$.
7. $\mathcal{KB} \setminus \{(\text{A.4})\}$.
8. $\mathcal{KB} \setminus \{(\text{A.5})\}$.
9. $\mathcal{KB} \setminus \{(\text{A.1})\}$. This is a maximal privacy preserving knowledge base and we are done.

In total, we needed nine calls to `isPP`.

4. Complexity

In this section we establish basic results about the complexity of the knowledge base distortion problem. We refer, for instance, to [9] for a detailed introduction to the complexity theoretic notions we use here.

Definition 6. The *underlying decision problem of the distortion problem* for a knowledge base $\mathcal{KB}$ is to decide whether there exists a solution $\mathcal{KB}'$ of the distortion problem for $\mathcal{KB}$ such that $\mathsf{card}(\mathcal{KB}') > k$ for a natural number k.

In the following we will only talk about *our underlying decision problem* when we mean the *underlying decision problem of the distortion problem*. Looking at our algorithms we see that to solve the underlying decision problem for a knowledge base $\mathcal{KB}$ and a natural number k, it is enough to call `doBoundedSearch` with the arguments $(\mathcal{KB}, k, \emptyset)$. If that call returns a knowledge base, then the answer to the decision problem is true; if it returns null, then the answer is false.

The consequence relation Con is an important ingredient in the definition of the distortion problem and its underlying decision problem. Thus the

complexity of Con contributes essentially to the overall complexity of our underlying decision problem. We will first abstract away from this by assuming that we have an oracle at hand which decides the Con relation. Then the following non-deterministic procedure solves our underlying decision problem:

1. guess a subset $\mathcal{KB}'$ of $\mathcal{KB}$,
2. if $\mathsf{card}(\mathcal{KB}') \leq k$ then reject,
3. if $\mathcal{KB}'$ preserves privacy with respect to $\mathcal{SC}$ then accept else reject.

Observe that all three steps of this procedure can be performed in polynomial time (the third step makes $\mathsf{card}(\mathcal{SC})$ many calls to the oracle). Thus we have the following result.

Theorem 7. *Assume that we are given an oracle L that decides the* Con *relation. Then our underlying decision problem belongs to* NP^L.

Let us now look at the complexity of concrete instances of the distortion problem. We begin with a setting like in Example 2. $\mathcal{L}_A$ and $\mathcal{L}_S$ are sets of classical propositional formulas. For a knowledge base $\mathcal{KB}$ and a secret S, the relation $\mathsf{Con}(\mathcal{KB}, S)$ is defined to hold if and only if $\mathcal{KB} \cup P \models S$ where P also is a set of classical propositional formulas that models the a priori knowledge. In this case the Con relation belongs to coNP and we get the following corollary.

Corollary 8. *Assume that* Con *belongs to* coNP. *Then our underlying decision problem belongs to* Σ_2^p.

Another important case for applications are very expressive description logics. For such logics, deciding Con has a high complexity which then determines the complexity of our underlying decision problem. Consider, for example, the description logic $\mathcal{SHIQ}$, see [12], which includes transitive and inverse roles, role hierarchies, and qualified number restrictions. Deciding the Con relation for this logic is an ExpTime problem. Therefore, we immediately get the following corollary about the distortion problem for $\mathcal{SHIQ}$.

Corollary 9. *Assume that* Con *belongs to* ExpTime. *Then our underlying decision problem belongs to* ExpTime.

At the other end of the complexity spectrum are application where the Con relation is decidable in polynomial time. Let us mention, for example, lightweight description logics that recently have been developed for data intensive applications. These logics are not as expressive as other description logics but they exhibit an excellent (at most polynomial time) run-time behavior. There are two important families of such logics: the EL family [2] which still can model several medical ontologies and the DL-Lite family [11] which allows for a separation of TBox and ABox reasoning.

Corollary 10. *Assume that* Con *belongs to* P. *Then our underlying decision problem belongs to* NP.

5. Heuristic

The algorithm for `doBoundedSearch` performs well if it finds the 'right' elements to remove from the knowledge base as early as possible. That is it has to make a clever choice with which element $a \in B$ it will start in the forall loop. So far there is no heuristic built into the algorithm that supports this choice: the loop may iterate in any order through B. Actually, there is no information available to make this an informed choice that selects the 'right' elements early. However, depending on the implementation of the decision procedure for Con, there exists a simple heuristic that can be added to our algorithm. We only need that if $\mathsf{Con}(\mathcal{KB}, S)$ holds, then a call to $\mathsf{Con}(\mathcal{KB}, S)$ returns a small subset $\mathcal{KB}'$ of $\mathcal{KB}$ such that already $\mathsf{Con}(\mathcal{KB}', S)$ holds.

This is the case, for example, if the implementation of Con performs a proof search. That means a call to $\mathsf{Con}(\mathcal{KB}, S)$ tries to construct a derivation of S from $\mathcal{KB}$. If $\mathsf{Con}(\mathcal{KB}, S)$ holds, it can provide a list of elements $a_1, \ldots, a_m$ of $\mathcal{KB}$ that were actually used in the derivation of S. Since we are looking for a subset $\mathcal{KB}''$ of $\mathcal{KB}$ such that $\mathsf{Con}(\mathcal{KB}'', S)$ does not hold anymore, it seems to be a good choice to remove one of the a_i above from $\mathcal{KB}$ in order to construct a candidate for $\mathcal{KB}''$. If we removed an element b different from $a_1, \ldots, a_m$, then the derivation of S would still be possible, that is $\mathsf{Con}(\mathcal{KB} \setminus \{b\}, S)$ holds. Of course, removing an element a_i does not guarantee that $\mathsf{Con}(\mathcal{KB} \setminus \{a_i\}, S)$ does not hold since there may be other derivations of S from $\mathcal{KB}$ that do not make use of a_i.

We are not only interested in preserving one secret but a whole set of secrets $\{S_1, \ldots, S_n\}$. Thus in Algorithm 3, we may not only check $\mathsf{Con}(A, S_i)$ until privacy is violated; instead we can check $\mathsf{Con}(A, S_i)$ for all $1 \leq i \leq n$ and keep track of which elements of A are used most often to construct derivations of secrets. If we build C by removing an element a_i of A that has been used in several derivations, then chances are good that C will preserve several secrets. Hence, Algorithm 3 should not return a boolean value but additionally also a priority queue of elements of A that were used to build the derivations of the secrets. For that queue, the more derivations an element as been used in, the higher priority it receives. The forall loop in Algorithm 2 can make use of this queue to start with an element of maximal priority.

We will now present pseudo code for a variant of `isPP` that returns such a priority queue. To do so we assume that the function $\mathsf{Con}(A, S)$ returns a pair $(result, list)$ where $result$ is true if and only if S is derivable from A, and additionally, if S is derivable, then $list$ contains those elements of A that have been used in the derivation of S. For instance, $\mathsf{Con}(A, S)$ returns $(\mathsf{true}, \emptyset)$ if S follows from the a priori knowledge alone.

Example 11. Consider the knowledge base $\mathcal{KB}$ and the set of secrets $\mathcal{SC}$ from Example 4. A call to $\texttt{isPP2}(\mathcal{KB})$ first tries to derive $\mathsf{Patient1} \rightarrow \mathsf{brokenLeg}$ from $\mathcal{KB}$. It will find such a derivation using (A.1), (A.2), and (A.3). Then the algorithm tries to derive $\mathsf{Patient2} \rightarrow \mathsf{highCosts}$. Again, it will find such a derivation using (A.1), (A.4), and (A.5). Therefore, the returned queue

Algorithm 4 `isPP2`(A)

Require: A is a knowledge base
Ensure: returns $(\mathsf{true}, \emptyset)$ if A preserves privacy with respect to $\mathcal{SC}$
Ensure: returns $(\mathsf{false}, queue)$ otherwise where $queue$ contains elements of A that have been used in derivations of secrets

```
privacyPreserving ← true
queue ← ∅
for all a ∈ A do
  numberOfUses[a] ← 0
end for
for i := 1 ... card(SC) do
  (result, list) ← Con(A, SC_i)
  if result then
    privacyPreserving ← false
    for all a ∈ list do
      numberOfUses[a] ← numberOfUses[a] + 1
    end for
  end if
end for
for all a ∈ A do
  if numberOfUses[a] > 0 then
    add a to queue with priority numberOfUses[a]
  end if
end for
return (privacyPreserving, queue)
```

contains (A.1) with priority 2 as well as (A.2), (A.3), (A.4), and (A.5) all with priority 1.

Hence the next try for a privacy preserving knowledge base will be the knowledge base $\mathcal{KB}' = \mathcal{KB} \setminus \{(\text{A.1})\}$ since (A.1) is the element with highest priority in the queue. As shown before, $\mathcal{KB}'$ is indeed privacy preserving and we are done with two calls to `isPP2` which is of course a great improvement compared to nine calls in Example 5.

The example shows that this heuristic considerably improves the performance of the search algorithm. There is also a simpler heuristic available which works as follows. The function `isPP3` works like `isPP`. That means, using a while loop it looks for a secret that can be derived. When the first such secret is found, it ends the loop and returns false. Additionally, and like `isPP2`, it also returns a list of elements of the knowledge base that have been used to derive the secret. However, since it stops when the first derivable secret is found, it will not return priorities. The search algorithm that calls `isPP3` will then remove one of these returned elements from the knowledge base to obtain the next candidate for a privacy preserving knowledge base.

On the one hand this variant of the search algorithm will need more calls to `isPP3` than it would need calls to `isPP2` since we do not have the priorities to choose a 'best' element. On the other hand `isPP3` makes less calls to Con since it does not test all the secrets.

To compare these two heuristics we made some example runs on a prototype implementation where the knowledge base and the secrets are propositional formulas and the consequence relation is the one from classical propositional logic. We are only interested in the number of calls to `isPP` and to Con but not in the actual run time. Therefore, we employed a straightforward implementation with no optimizations and we tested only very small knowledge bases (50 formulas) that were randomly generated. For different sizes of the solution we present a representative example in the following table where *Size* is the size of solution of the distortion problem and `isPP` is the number of calls to `isPP` (similar for Con, `isPP2`, and `isPP3`). We see that the more complex heuristic performs better than the simple one as soon as more than one element have to be removed from the knowledge base in order to obtain a solution.

	Uninformed Search		Heuristic		Simple Heuristic	
Size	`isPP`	Con	`isPP2`	Con	`isPP3`	Con
46	1515534	9206278	12	300	20877	126066
47	52847	245677	8	200	2546	11564
48	3568	14993	4	100	52	231
49	679	10685	2	50	2	40

6. Conclusion and further work

We investigated the distortion problem which consists in generating a privacy preserving knowledge base from a give unsecure knowledge base. We gave a universal definition of that problem and first basic algorithms to solve it where our definition and the algorithms are independent of the used knowledge representation language. From our algorithms we derived upper bounds on the complexity of the distortion problem both for the general case and for particular logics. We also provided a heuristic to improve the average runtime and illustrated its impact on the performance.

Future work will include the development of specialized algorithms for particular knowledge representation languages. The aim is to make use of optimized reasoning techniques that are available for a given logic to find high-performance algorithms for the distortion problem of that logic.

Another direction of future research is to consider more general versions of the distortion problem. We required a solution to be a maximal subset with respect to cardinality. This is a form of *minimal change*, a notion that also occurs in the theory of belief revision and updates. However, minimal change means, in general, that as many consequences as possible should persist. To achieve this, it can be necessary that elements not only have to be removed

from a knowledge base but also new elements have to be added. As an example, consider again a knowledge base about patients and diseases where the knowledge base should guarantee the following privacy condition which can be seen as a form of 2-anonymity [13]: it should be kept secret who exactly has a given disease but a user may know two patients such that at least one of them has the disease. In our running example we have that $\mathcal{KB}$ (together with the a priori knowledge) implies that

$$\mathsf{Patient1} \rightarrow \mathsf{brokenLeg} \tag{1}$$

which should be kept secret. We find a maximal privacy preserving knowledge base $\mathcal{KB}'$ by removing assertions from $\mathcal{KB}$ until (1) does not follow anymore. Since we are only interested in 2-anonymity, we may then again add some assertions to $\mathcal{KB}'$ like $\mathsf{Patient1} \vee \mathsf{Patient3} \rightarrow \mathsf{brokenLeg}$ which was a consequence of the original knowledge base $\mathcal{KB}$ but not of $\mathcal{KB}'$. Of course, the resulting knowledge base is not a subset of $\mathcal{KB}$ anymore; but with respect to consequences it is closer to $\mathcal{KB}$ than any of its privacy preserving subsets.

References

[1] R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu. Hippocratic databases. In *Proc. of 28th VLDB Conference*, 2002.

[2] F. Baader, S. Brandt, and C. Lutz. Pushing the $\mathcal{EL}$ envelope. In L. P. Kaelbling and A. Saffiotti, editors, *IJCAI-05, Proceedings*, pages 364–369, 2005.

[3] J. Biskup and P. A. Bonatti. Controlled query evaluation for enforcing confidentiality in complete information systems. *Int. J. Inf. Sec.*, 3(1):14–27, 2004.

[4] J. Biskup and P. A. Bonatti. Controlled query evaluation for known policies by combining lying and refusal. *Ann. Math. Artif. Intell.*, 40(1-2):37–62, 2004.

[5] J. Biskup and P. A. Bonatti. Controlled query evaluation with open queries for a decidable relational submodel. *Annals of Mathematics and Artificial Intelligence*, 50(1-2):39–77, 2007.

[6] J. Biskup and T. Weibert. Keeping secrets in incomplete databases. *Int. J. Inf. Sec.*, 7(3), 2008.

[7] J. Biskup and L. Wiese. Preprocessing for controlled query evaluation with availability policy. *Journal of Computer Security*, 16(4):477–494, 2008.

[8] P. A. Bonatti, S. Kraus, and V. s. Subrahmanian. Foundations of secure deductive databases. *Transactions on Knowledge and Data Engineering*, 7(3):406–422, 1995.

[9] D. P. Bovet and P. Crescenzi. *Introduction to the Theory of Complexity*. Prentice Hall, 1994.

[10] Bundesamt für Statistik. Medizinische Statistik der Krankenhäuser.

[11] D. Calvanese, G. Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Tractable reasoning and efficient query answering in description logics: The DL-Lite family. *J. Autom. Reason.*, 39(3):385–429, 2007.

[12] I. Horrocks, U. Sattler, and S. Tobies. Practical reasoning for very expressive description logics. *Logic Journal of the IGPL*, 8(3), 2000.

[13] P. Samarati and L. Sweeney. Generalizing data to provide anonymity when disclosing information (abstract). In *PODS*, page 188, 1998.

[14] S. Schlobach and R. Cornet. Non-standard reasoning services for the debugging of description logic terminologies. In G. Gottlob and T. Walsh, editors, *IJCAI*, pages 355–362, 2003.

[15] G. L. Sicherman, W. De Jonge, and R. P. Van de Riet. Answering queries without revealing secrets. *ACM Trans. Database Syst.*, 8(1):41–59, 1983.

[16] K. Stoffel and T. Studer. Provable data privacy. In K. Viborg, J. Debenham, and R. Wagner, editors, *DEXA 2005*, volume 3588 of *LNCS*, pages 324–332. Springer, 2005.

Thomas Studer
Universität Bern
Institut für Informatik und angewandte Mathematik
Neubrückstrasse 10 CH 3012 Bern
Switzerland
e-mail: `tstuder@iam.unibe.ch`