INFORMATICA, 2003, Vol. 14, No. 3, 277–288
© 2003 *Institute of Mathematics and Informatics, Vilnius*

An Algorithm for Protecting Knowledge Discovery Data

Boštjan BRUMEN, Izidor GOLOB, Tatjana WELZER, Ivan ROZMAN
University of Maribor, Faculty of Electrical Engineering and Computer Science
Smetanova 17, Si-2000 Maribor, Slovenia
e-mail: {bostjan.brumen, izidor.golob, welzer, i.rozman}@uni-mb.si

Marjan DRUŽOVEC
University of Maribor, Faculty of Mechanical Engineering
Smetanova 17, Si-2000 Maribor, Slovenia
e-mail: marjan.druzovec@uni-mb.si

Hannu JAAKKOLA
Tampere University of Technology, Pori Technology and Economics
PO BOX 300, Fi-28101 Pori, Finland
e-mail: hj@pori.tut.fi

Received: March 2003

Abstract. In the paper, we present an algorithm that can be applied to protect data before a data mining process takes place. The data mining, a part of the knowledge discovery process, is mainly about building models from data. We address the following question: can we protect the data and still allow the data modelling process to take place? We consider the case where the distributions of original data values are preserved while the values themselves change, so that the resulting model is equivalent to the one built with original data. The presented formal approach is especially useful when the knowledge discovery process is outsourced. The application of the algorithm is demonstrated through an example.

Key words: data protection algorithm, disclosure control, data mining, knowledge discovery, data security.

1. Introduction

The fundamentals for data protection have been around for almost 40 years, coexisting with the fundamentals of data storage, and have become very solid since the introduction of relational database systems. The new technologies, such as knowledge discovery using data mining, need to be used with caution when the data are sensitive. Today's systems are interconnected; the data are viewed as company's important assets. They can be processed, analysed and converted into information upon which the management can make tactic and strategic decisions. Once the data are intelligently analysed and presented, they become a valuable resource to be used for a competitive advantage (Hedberg, 1995). In the last decade, the companies have recognized the value in data, especially with the introduction of the knowledge discovery process in 1991 (Shapiro, 1991), and later even more with the success stories of data mining. When it came to start using the data for data mining, the larger companies were able to find the resources for its implementation. Namely, the sheer volume of the data prevents from conducting human-driven analyses. Machine support and intelligent data analysis are definitely required. The smaller enterprises do not have enough (if any) expertise for doing the data analyses, although they do have good domain knowledge and understand their data structures much better. They have two choices: not mining the data at all or doing it with help from outside. Not doing data mining is sometimes not an option due to competitive reasons; the outsourced data mining poses a potential security threat to data (Clifton, 1996). Due to the requirements of most of today's analytical tools, data need to be prepared in a flat file. The reason is that many tools are developed to avoid database management systems overhead, and to assure compatibility across database and operating systems platforms. As such, data are much more easily copied and distributed. Therefore, mechanisms that will adequately protect the knowledge discovery data are needed, especially when the knowledge discovery process is outsourced.

Related work. One approach to protect the data is to use the techniques developed for the statistical databases, where the goal is to provide statistical data (e.g., sums, counts, averages, minimums, maximums) without exposing sensitive individual records (Adam, 1989). Two major systems, namely μ-Argus (Hundepool, 1996) and Datafly (Sweeney, 1997) have been developed. In these cases, the sensitive data values were generalized (Samarati, 2001) or not disclosed, respectively. In the data analyses world, we cannot have data that have been distorted or changed. For example, a decision rule on generalized and not disclosed data would read `IF body_temperature = ‘not released’` $\wedge$ `pressure = 'not released'` $\wedge$ `AGE = 'young'` $\wedge$ `zip_code ='9xxxx' THEN illness = 7`. Obviously, managers would find such form of discovered knowledge useless.

Another approach is to modify the real value of an attribute using value-class membership technique or value distortion (Agrawal, 2000), and trying to reconstruct the original distribution as close as possible (Agrawal, 2001). In the first case, the values are partitioned into a set of disjoint, mutually exclusive classes, while in the second case the values are slightly changed, namely the value $x_i + r$ is used instead of real value x_i; the r is a random value drawn from some distribution. The authors in (Agrawal, 2000) show that the approach can be used for numeric attributes (only) and for constructing a classifier. Their research focuses on the case where the data are distorted. They expect that the data are (deliberately) changed at the entry point into a system, namely by a (web) user. In many cases, the models built are very sensitive to distorted values. For example, a decision rule on distorted data may read `IF body_temperature = 38` $\wedge$ `pressure = 220 THEN illness = 7` instead of `IF body_temperature = 37` $\wedge$ `pressure = 190 THEN illness = 7`. An action based on wrong decision rule can have fatal consequences, especially where the values are very sensitive to small changes. The works in (Agrawal, 2000) and (Agrawal, 2001) are orthogonal to our approach and can be used complementarily with ours if needed.

Paper contribution. The paper addresses the problem where data need to be protected and still the data modelling process needs to be allowed. We propose a solution where the distributions of original data values are preserved while the values themselves change, so that the resulting model is equivalent to the one built with original data. The resulting model is useless unless the transformation functions are known and applied. The data owner, not the data modeller, knows the transformation functions. The presented formal approach is especially useful when the knowledge discovery process is outsourced.

Paper organization. We present the algorithm for protecting the data while at the same time enabling the construction of data models in Section 2. We present an experimental evaluation of the algorithm in Section 3. We conclude with a summary and outline our future work in Section 4.

2. An Algorithm for Protecting Data

We continue the work presented in (Brumen, 2002), where we identified the functions that transform the values and at the same time do not distort the distributions and can as such be used to protect the data.

In this paper we propose an algorithm that will use the data protection functions, not only for data values, but also for their structure. In our approach, we transform the (relational) database that is to be exported to the outside world. Importantly, the transformations are to be performed on both, the data structure and the data values. The reason to transform the data values is obvious; the data structure is transformed so that the semantics about the data is hidden to the outside world. Furthermore, a model built using transformed data is equivalent to the one using original data. Hence, the "transformed" model can be translated to the model built on original data using the same model-building technique (Fig. 1).

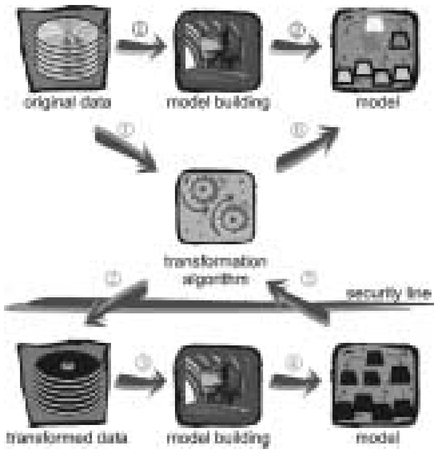

Fig. 1. Knowledge discovery data protecting algorithm.

In the Section 2.1, we briefly present the notation and basic definitions for relational data model RDM. We adopt the terminology from (Helman, 1994) and (Elmasri, 1999), specially the abstract data type (ADT) and the relational data model (RDM) that is based on it. The ADT search table and RDM operators are not described here; for formal definitions, which are beyond the scope of this paper, refer to (Helman, 1994). In the Section 2.2, following the definitions, we specify the algorithm.

2.1. *Definitions*

Suppose we have a *relational database* $\mathcal{R}$, which is actually a set of *relational tables*, $\mathcal{R} = \{T_1, \ldots, T_i, \ldots, T_I\}$. Each relational table (also called search table) T_i consists of a *table scheme* TS_i, which is invariant over the life of the table and, at any point of time, a set of *tuples* consistent with the scheme. Such a set of tuples is called the current *value* or *instance* of table T_i. The *relational database schema* $\mathcal{RS}$ is described by a set of *relational table schemas*, $\mathcal{RS} = \{TS_1, \ldots, TS_i, \ldots, TS_I\}$. A table schema of T_i, TS_i, is described by a finite set of *attributes*, $TS_i = \{A_{i1}, \ldots, A_{in} \ldots, A_{iN}\}$. Each attribute A_{in} is the name of a role played by some domain D_{in} in the relation schema TS_i. We assume that any attribute name A_{in} appears only once within table schema $TS_i.D_{in}$ is called the domain of A_{in} and is denoted by $D_{in} =$ dom(A_{in}). A *domain* is simply a set of values and can be finite or infinite. In a relational database, we have a set of domains, $D = \{D_1, \ldots, D_k, \ldots, D_K\}$. A *table* T_i consists of a table scheme TS_i and a set of N-tuples t_i, $t_i = \{t_{i1}, \ldots, t_{im}, \ldots, t_{iM}\}$. Each N-tuple t_{im} (belonging to the search table T_i) is an ordered list of N values, $t_{im} = \langle v_{im1}, \ldots, v_{imn}, \ldots, v_{imN}\rangle$, where each value v_{imn} is an element of dom(A_{in}), or is a special null value. The nth value of the mth tuple (t_{im}) of search table T_i, which corresponds to the attribute A_{in}, is referred to as $v_{imn} = t_{im}[A_{in}]$.

In the following three definitions, we set up a formal framework for the data-protecting algorithm. In Lemma 1, we claim that the algorithm is reversible and prove it in Proofs 1–3. The algorithm needs to be reversible so that the owner of the data can decode any results from the data mining process back to the readable form.

DEFINITION 1. Let D_{in} and D^*_{in} be sets. A *function* from D_{in} into D^*_{in} is a subset F of $D_{in} \times D^*_{in}$ if for each element $a \in D_{in}$ there is exactly one element $b \in D^*_{in}$ such that $(a, b) \in F$. Set D_{in} corresponds to domains of relation schema TS_i.

DEFINITION 2. Let *D** be a set of domains, such that $D^* = \{D^*_{in} | |D^*_{in}| = |D_{in}|\}$. Let $D_{in} \rightarrow D^*_{in}$ be a function, and $F^{Di} = \{f_{in}()\}$ be a set of transformations f_{in}. *Transformation* f_{in} is said to transform D_{in} onto D^*_{in}, if

1. $\forall b \exists a (f_{in}(a) = b)$;
2. $f_{in}(a_1) = f_{in}(a_2) \Leftrightarrow a_1 = a_2$.

DEFINITION 3. The function $\mathcal{N}$: *String* $\rightarrow$*String* is called a *rename operation* if:

1. $\forall b \exists a (\mathcal{N}(a) = b)$;
2. $\mathcal{N}(a_1) = \mathcal{N}(a_2) \Leftrightarrow a_1 = a_2$.

This way we defined a function that transforms table and attribute names into new values.

2.2. *Specification of the Algorithm*

Using the above definitions, we specify the Algorithm $\mathcal{A}$, which transforms a database $\mathcal{R}$ into $\mathcal{R}^*$. The database $\mathcal{R}^*$ is protected and can be exported to the outside world.

Algorithm $\mathcal{A}$:

```
FOR i=1 TO I DO
  BEGIN
    CREATE $(\mathcal{N}(T_i)\ (\mathcal{N}(A_{i1}),\ldots,\mathcal{N}(A_{in}),\ldots,\mathcal{N}(A_{iN}))$
    ADD_SCHEME $(\mathcal{N}(TS_i), \mathcal{RS}^*)$
    FOR m=1 to M DO
        ADD $(\mathcal{N}(T_i), < f_{i1}(v_{im1}),\ldots,f_{in}(v_{imn}),\ldots,f_{iN}(v_{imN}) >)$
    END
```

The relation schema of database $\mathcal{R}^*$ is essentially the same as the one of the $\mathcal{R}$. That is, the number of the relation table schemas is the same and each table schema in $\mathcal{R}^*$ has the same number of attributes as the corresponding table schema in schema $\mathcal{R}$. Such a database is needed so that the instances are easily transformed from database $\mathcal{R}$ into database $\mathcal{R}^*$. The Algorithm $\mathcal{A}$ defines the transformation.

Lemma 1. *Algorithm $\mathcal{A}$ is reversible.*

Proof. Algorithm $\mathcal{A}$ is reversible if functions $\mathcal{N}$ and $\{f_{1n},\ldots,f_{in},\ldots,f_{In}\}$ are bijective. If they are bijective, the inverses exist. If the inverses exist, the algorithm has a reverse, $\mathcal{A}^{-1}$.

Algorithm $\mathcal{A}^{-1}$:

```
FOR i=1 TO I DO
  BEGIN
    CREATE $(\mathcal{N}^{-1}(T_i^*)\ (\mathcal{N}^{-1}(A_{i1}^*),\ldots,\mathcal{N}^{-1}(A_{in}^*),\ldots,\mathcal{N}^{-1}(A_{iN}^*))$
    ADD_SCHEME $(\mathcal{N}^{-1}(TS_i^*), \mathcal{RS})$
    FOR m=1 to M DO
        ADD $(\mathcal{N}^{-1}(T_i^*), < f_{i1}^{-1}(v_{im1}^*),\ldots,f_{in}^{-1}(v_{imn}^*),\ldots,f_{iN}^{-1}(v_{imN}^*) >)$
    END
```

Therefore, we first prove the next lemma stating that the functions are bijective. Since the functions $\mathcal{N}$ and $\{f_{1n},\ldots,f_{in},\ldots,f_{In}\}$ are defined exactly in the same manner (see Definition 2 and Definition 3), we use a generic function f and prove the following lemmas using it.

Lemma 2. *Function f is bijective.*

Function f is bijective if it is injective and surjective.

A function f is said to be injective, if and only if whenever $f(p) = f(q) \Rightarrow p = q$. Suppose we have two values, $p, q \in D$. Since $f(p) = f(q) \Rightarrow \{p^*|p^* = f(p)\} = \{q^*|q^* = f(q)\} \Rightarrow p^* = q^*$. Since by Definition 2 $f(a) = f(b)$ when $a = b \Rightarrow p = q$.

A function f is said to be surjective, if and only if for any $q^* \in D^*$, there exists an element $p \in D$ such that $f(p) = q^*$.

Suppose we have $q^* \in D^*$. Each q^* is calculated as $q^* = f(p)$. From Definition 2 we have that for function f, for each b exists an a such that $f(a) = b$. By declaring $s^* = b$ it is evident that there must exist a p, so that *f(p)* = $q^* \Rightarrow \forall\, q^*\colon\ \exists\, p(f(p) = q^*)$.

Lemma 3. *Function f has an inverse, f^{-1}, such that $f^{-1}(r^*) = r$, where $f(r) = r^*$.*

Proof. Let f be bijection. Then exists one and only one bijection f^{-1} that implies

1. $f^{-1}(f(p)) = p$ for $\forall p \in D$.
2. $f(f^{-1}(q)) = q$ for $\forall q \in D^*$.

That unique bijection f^{-1} is called the inverse function of f.

We first prove that $f^{-1}(f(p)) = p$ for $\forall p \in D$.

Since f is injection: $f(p_1) = f(p_2) \Rightarrow p_1 = p_2$. Since f^{-1} is a function: $f(p_1) = f(p_2) \Rightarrow f^{-1}(f(p_1)) = f^{-1}(f(p_2))$, i.e., $p_1 = p_2$.

Next, we prove that $f(f^{-1}(q)) = q$ for $\forall q \in D^*$.

Since f is surjection: for any $q \in D^*$ there exists $p \in D$ such that $f(p) = q$ and since f^{-1} is a function: $p = f^{-1}(q)$ is defined for every $q \in D^*$.

Thus, $f(p) = f(f^{-1}(q)) = q$.

The owner of the database $\mathcal{R}$ can apply the algorithm $\mathcal{A}$ first, and give to the third party a transformed database $\mathcal{R}^*$. The functions $\mathcal{N}$ and $\{f_{1n}, \ldots, f_{in}, \ldots, f_{In}\}$ are kept secret. The person trying to find out what the data really mean has an obstacle because the semantics of the data is not known. That is, statistical approach can be tried, but if the number of attributes is high (which is usually the case), the combinatorial explosion of the possible answers makes the statistical attack very difficult. If F^{Di} is carefully chosen the attack becomes infeasible. The functions that can be chosen are strong encryption algorithms or other functions that preserve statistical properties of data (see (Adam, 1989; Willenborg, 1996)). The advantage is that the F^{Di} can easily be chosen so that it corresponds to the value of data to be protected.

3. Using the Algorithm

For clearer impression of the presented algorithm, let us take a closer look at an example of a database with three tables. We transform them using the above algorithm.

Suppose we have a set of domains D:

```
D={Integer, String, Boolean, Date},
```

and a relational database schema $\mathcal{RS} = \{TS_i\}$:

```
DB={PATIENT, ILLNESS, PATIENT_HISTORY},
```

where each relation schema is denoted as

```
PATIENT(P_ID, Fname, Lname, Zip, City, Age);
PATIENT_HISTORY(P_ID, I_ID, Date_Discovered);
ILLNESS(I_ID, Name, Fatality);
```

and each attribute has the following domain:

```
PATIENT:
Dom(P_ID)=Integer, Dom(Fname)=String, Dom(Lname)=String,
Dom(Zip)=Integer, Dom(City)=String, Dom(Age)=Integer;
Dom(Smoking)=Boolean;

ILLNESS:
Dom(I_ID)=Integer, Dom(Name)=String, Dom(Fatality)=Integer;

PATIENT_HISTORY:
Dom(P_ID)=Integer, Dom(I_ID)=Integer, Dom(Date_Discovered)=Date.
```

We have a set of relations (instances) of relation schemas, presented in a tabular form:

`patient(PATIENT):`

P_ID	*Fname*	*Lname*	*Zip*	*City*	*Age*	*Smoking*
1	Johann	Wolf	10000	Berlin	65	Y
2	Juha	Tantu	20000	Helsinki	77	N
3	Georgos	Ponulis	30000	Athens	46	Y

`illness(ILLNESS):`

I_ID	*Name*	*Fatality*
1	Angina Pectoris	1
2	Lung Cancer	10
3	Osteoporosis	2
4	Sarcoidosis	4

`patient_history(PATIENT_HISTORY):`

P_ID	*I_ID*	*Date_discovered*
1	1	10-Jan-1986
1	2	12-Feb-1997
2	4	16-Mar-2001
2	3	18-Apr-2000
3	1	19-May-2001
3	2	22-Jun-2001
3	3	25-Jul-2002

Suppose an outside entity does models-based data mining on the above tables. In models-based data mining, the goal is to make a model based on the underlying data. The models can be neural networks, decision trees, decision lists, association rules, or a set of clusters, to name only a few. The knowledge discovery system may come up with the following model (rule) using the above data: `IF patient.smoking='Y' THEN illness.name='Lung Cancer'`. As we can see, the parts of the rule actually correspond to a simple relational database with two instances:

`PATIENT`
Smoking
Y

`ILLNESS`
Name
Lung Cancer

If the knowledge discovery was run outside of the medical enterprise, the data could be compromised. The compromised data can be misused (e.g., sold to an insurance company). To prevent this, we can use the Algorithm $\mathcal{A}$. For example, when building a decision tree, the tree-building algorithm makes a branch based on some statistical properties of data, not on actual values or attribute names. For these reasons the actual data and their structure can be hidden, and the results will still be the same, as long as statistics within data is maintained.

Suppose we define bijective functions $\mathcal{N}$ and $\{f_{1n}, \ldots, f_{in}, \ldots, f_{In}\}$ as follows:

$$\begin{aligned}
\mathcal{N} &= f_{12} = f_{13} = f_{15} = f_{17} = f_{22} \\
&= \text{ASCII(name[1])}|\ldots|\text{ASCII(name[i]}|\ldots|\text{ASCII(name[length(name)])} \\
f_{11} &= f_{16} = f_{21} = f_{23} = f_{31} = f_{32} = 2x^3 + 3, \\
f_{14} &= 2(x/1000)^3 + 3, \\
f_{31} &= YYYY * 10000 + MM * 100 + DD.
\end{aligned}$$

The proof that they are bijective is beyond the purpose of this example. If we use the Algorithm $\mathcal{A}$, we get the following database $\mathcal{R}^*$:

`112097116105101110116(080065084073069078084): (corresponds to patient)`

80095073068	*70110097109101*	*76110097109101*	*90105112*	*67105116121*	*65103101*	*8310911110 7105110103*
5	74111104097110110	87111108102	203	66101114108105110	549253	89
19	74117104097	84097110116117	803	72101108115105110107105	913069	78
57	71101111114103111115	80111108105115	1803	65116104101110115	194675	89

`105108108110101115115(073076076078069083083): (corresponds to illness)`

73095073068	*78097109101*	*70097116097108105116121*
5	65110103105110097032080101099116111114105115	5
19	76117110103032067097110099101114	1003
57	79115116101111112111114111115105115	19
131	83097114099111105100111115105115	131

```
11209711610510111011609510410511511611111141210
(80065084073069078084095072073083084079082089):
(corresponds to patient_history)
```

80095073068	*73095073068*	*68097116101095100105115099111118101114101100*
5	5	19860110
5	19	19970212
19	131	20010316
19	57	20000418
57	5	20010519
57	19	20010622
57	57	20010725

We may freely give out the database $\mathcal{R}^*$. The data discovery process will come up with the following rule:

```
IF 112097116105101110116.83109111107105110103=089
THEN 105108108110101115115.78097109101 = 651101031
05110097032080101099116111114105115.
```

This corresponds to a simple relational database with two instances:

`112097116105101110116`
83109111107105110103
89

`105108108110101115115`
78097109101
65110103105110097032080101099116111114105115

The database can be converted using the $\mathcal{A}^{-1}$ algorithm. For this we need to find inverses to the functions $\mathcal{N}$ and $\{f_{1n}, \dots, f_{in}, \dots, f_{In}\}$, which are the following:

$$\begin{aligned}
\mathcal{N}^{-1} &= f_{12}^{-1} = f_{13}^{-1} = f_{15}^{-1} = f_{17}^{-1} = f_{22}^{-1} \\
&= \text{CHR(name[1])}|\dots|\text{CHR(name[i]}|\dots|\text{CHR(name[length(name)])}, \\
f_{11}^{-1} &= f_{16}^{-1} = f_{21}^{-1} = f_{23}^{-1} = f_{31}^{-1} = f_{32}^{-1} = \sqrt[3]{\frac{x-3}{2}}, \\
f_{14}^{-1} &= 1000 \times \sqrt[3]{\frac{x-3}{2}}, \\
f_{31}^{-1} &= TO\_DATE(xDIV10000, (x - (xDIV10000) * 10000), xMOD100).
\end{aligned}$$

Using the inverses and the $\mathcal{A}^{-1}$ algorithm, the values decode to the same rule as above: `IF patient.smoking='Y' THEN illness.name='Lung Cancer'`. The parts of the rule correspond to the following database instance:

`PATIENT`
Smoking
Y

`ILLNESS`
Name
Lung Cancer

4. Conclusions and Future Work

In this paper we studied the feasibility of technically protecting data before any knowledge discovery process takes place. The premise was that the values remain unchanged and the statistics inside the data remain intact. We presented an algorithm for protecting the data when doing model-driven data mining, especially when the data are transferred to an insecure environment. The algorithm is reversible, thus allowing the results of the models being translated back to the readable form, but only by the owner of the data. The algorithm was developed so that the data remain unchanged (non-distorted) and visible (not hidden, completely disclosed). Furthermore, the statistics inside the data remain the same. This way the modelling algorithm performs equally well on the protected as on the non-protected data. The attacker is faced with two obstacles. First, the data semantics is hidden. If a statistical attack is performed, all attributes have to be checked against all distributions for all possible matches. With a growing number of attributes, this becomes infeasible, if not impossible. Second, the transformation functions need to be broken as well. If strong encryption functions or other functions that preserve statistics are used, the attack may take a lot of time. Finally, the approach can be used with other techniques as well (such as generalization and non-disclosure on one hand and value distortion on the other), if the data-modelling task permits so. In our future work, we plan to empirically evaluate strong transformation functions to be used in our algorithm.

References

Adam, N.R., and J.C. Wortman (1989). Security-control methods for statistical databases: a comparative study. *ACM Computing Surveys*, **21**(4), 515–556.

Agrawal, R., and R. Srikant (2000). Privacy-preserving data mining. In *Proceedings of the 2000 ACM SIGMOD international conference on Management of Data*. Dallas, Texas.

Agrawal, D., and Ch.C. Aggrawal (2001). On the design and quantification of privacy preserving data mining algorithms. In *Proceedings of the ACM PODS'01 Conference*. Santa Barbara, California.

Brumen, B., I. Golob, T. Welzer, M. Družovec, I. Rozman and H. Jaakkola (2002). Data protection for outsourced data mining. *Informatica*, **26**(2), 205–210.

Clifton, Ch., and D. Marks (1996). Security and privacy implications of data mining. In *Proceedings of the 1996 ACM SIGMOD Workshop on Data Mining and Knowledge Discovery*.

Elmasri, R.A., and S.B. Navathe (1999). *Fundamentals of Database Systems*. 3rd edition, Addison-Wesley Publishing, New York.

Hedberg, S.R. (1995). The data gold rush. *Byte Magazine*, **10**.

Helman, P. (1994). *The Science of Database Management*. Irwin Inc.

Hundepool, A.J., and L.C.R.J. Willenborg (1996). μ- and τ-argus: software for statistical disclosure control. In *Proceedings of 3rd International Seminar on Statistical Confidentiality*. Bled.

Samarati, P. (2001). Protecting respondents' intentities in microdata release. *IEEE Trans. on Knowledge and Data Engineering*, **13**(6).

Piatetsky-Shapiro, G., and W.J. Frawley (Eds.) (1991). *Knowledge Discovery in Databases*. AAAI/MIT Press, USA.

Sweeney, L. (1997). Guaranteeing anonymity when sharing medical data, the datafly system, proceedings. *Journal of American Medical Informatics Association*, Washington, DC, Hanley & Belfus Inc.

Willenborg, L.C.R.J., and T. De Waal (1996). Statistical disclosure control in practice. *LNS*, **111**, Springer–Verlag.

B. Brumen is a teaching assistant at Faculty of Electrical Engineering and Computer Science, University of Maribor. His research interests include data mining, data security, data analyses, and data reusability. As a member of Database Technologies Laboratory he actively participates in several international and national projects, related to data issues. He cooperates with researchers at Tampere University of Technology since 1999. The results were published at several international conferences and in journals.

I. Golob is a doctoral candidate at the Faculty of Electrical Engineering and Computer Science, University of Maribor. He earned his bachelor's degree in 1997 and master's degree in 2001, both from University of Maribor. His research interests include databases, information quality and data warehousing.

T. Welzer is an associate professor at Faculty of Electrical Engineering and Computer Science, University of Maribor. She graduated in 1984, received master's degree in computer science in 1989 and doctor's degree in 1995. Her research interests include data modelling, data technologies, data reuse and medical informatics.

I. Rozman is a professor at Faculty of Electrical Engineering and Computer Science, University of Maribor. He graduated in 1977 (electrical engineering), received master's degree (computer science) in 1980 and doctor's degree (computer science) in 1983. His research interests include integration of information systems, software metrics and systems quality.

M. Družovec is a teaching assistant at the Faculty of Mechanical Engineering, University of Maribor. He graduated in 1979, received master's degree in 1991 and doctor's degree in 1995 in computer science. His research interests include knowledge systems, knowledge discovery and system diagnostics.

H. Jaakkola is a professor of software engineering at Tampere University of Technology, director of Center of Software Expertise (CoSE) and head of the Regional Institute of Tampere University of Technology in Pori. His research interests cover software engineering (software process improvement and object technologies) and technology management (technology diffusion and transfer). Professor Hannu Jaakkola has received doctor's degree (engineering) at Tampere University of Technology in 1991, BSc (business economics) at University of Tampere in 1979 and master's degree (computer science) at University of Tampere in 1974.

Duomenų, naudojamų žinioms išskirti, apsaugos algoritmas

Boštjan BRUMEN, Izidor GOLOB, Tatjana WELZER, Ivan ROZMAN,
Marjan DRUŽOVEC, Hannu JAAKKOLA

Straipsnyje pateikiamas algoritmas, leidžiantis apsaugoti duomenis prieš pradedant juos analizuoti, siekiant sukurti modelius šių duomenų pagrindu. Nagrinėjama problema – ar galima kokybiškai analizuoti duomenis prieš tai juos apsaugojus siūlomu būdu? Tiriamas atvejis, kuomet koduotų duomenų pasiskirstymas išlieka toks pats, kaip ir pradinių, tuo tarpu kai jų reikšmės yra pakitę. Siekama, kad gautas modelis apsaugotų duomenų pagrindu būtų ekvivalentus gautam pagal realius duomenis. Pateikiamas algoritmo taikymo pavyzdys.