Data protection for outsourced data mining

Boštjan Brumen, Izidor Golob, Tatjana Welzer and Ivan Rozman
University of Maribor, Faculty of Electrical Engineering and Computer Science
Smetanova 17, Si-2000 Maribor, Slovenia
{bostjan.brumen | izidor.golob | welzer | i.rozman}@uni-mb.si
AND
Marjan Družovec
University of Maribor, Faculty of Mechanical Engineering
Smetanova 17, Si-2000 Maribor, Slovenia
marjan.druzovec@uni-mb.si
AND
Hannu Jaakkola
Tampere University of Technology, Pori School of Technology and Economics
PO BOX 300, Fi-28101 Pori, Finland
hj@pori.tut.fi

Keywords: data protection, data mining, outsourcing

Received: January 18, 2002

In the paper, we present data mining from the data protection point of view. In many cases, the companies have a lack of expertise in data mining and are required to get help from outside. In this case the data leave the organization and need to be protected against misuse, both legally and technically. In the paper a formal framework for protecting the data that leave the organization's boundary is presented. The data and the data structure are modified so that data modeling process can still take place and the results can be obtained, but the data content itself is hard to reveal. Once the data mining results are returned, the inverse process discloses the meaning of the model to the data owners. The approach is especially useful for model-based data mining.

1 Introduction

The traditional means for collecting the data were restricted to the use of our natural senses – and so were the means for storing the data. Then, people started to use other, more persistent means for data storage, such as skins, stones and much later, papyrus and paper. But only since the advent of electricity (or more specifically, electronics) the collection of data is no more restricted to human natural senses only.

Electronic equipment today, operative in such diverse fields as geology, business, astronomy, or medicine is capable of gathering vast amounts of data. It is to notice that the storage nowadays is affordable: in 1980, an IBM PC had a 5 MB hard drive and was priced at $3000, and in 1995, for the same price, it was equipped with a 1GB hard drive [Bigus, 1999]. Today (in 2002), an IBM PC at the same price (not adjusted for inflation!) is shipped with an 80 GB hard drive. In the first 15 years, the amount of disk storage available in a PC compatible computer increased over 200 times. From 1995 to today, the increase is almost 80-fold and cumulative factor of increase since 1980 is more than 16000. The situation is equally scaled with larger computers.

In other words, the amount of data storage (and consequently the amount of data actually stored) doubles roughly every 15 months, beating the famous Moore's law[1] by 3 months. The last boost was clearly powered by the wide use of the Internet. In the early 1990s, the computers have definitely moved data in paper form to on-line databases.

For the last 30 years, the data were mainly a by-product of daily operations of a company. In general, not much was used for analytical purposes. But, the data are every company's vital assets [Reinhardt, 1995]. Assets are useless, unless they are used in (strategic) business processes. Data are facts and as such do not contribute to a company's competitive advantage. Thus, data are useless if they only reside in a company – even worse, they are causing costs: maintenance, storage, and security, to mention only a few. For this reason, there is a need that data be processed, analyzed, and converted into information. Upon information, company's decision-makers can act and sustain competitive advantage. Thus, once data are intelligently analyzed and presented, they

[1] Dr. Gordon E. Moore stated in 1965 that the number of transistors per square inch on integrated circuits (and thus the processing power) doubles roughly every 18 months [Moore, 1965].

become a valuable resource to be used for a competitive advantage [Hedberg, 1995]. Data indeed represent a great potential.

Many companies have recognized the value in data and started to systematically collect and store the data. When it came to using the data for other purposes than daily operations, the traditional transactional systems became to fail answering the questions in reasonable time, simply because they were not designed for such queries. In early 1990s a new paradigm was being introduced: the data warehouses.

Today, a modern, efficient and effective information system consists of "two hearts" – a database and a data warehouse (Figure 1).

The transactional databases take care of daily transactions, such as "receive payment" or "ship goods". The data warehouse part is responsible for answering ad-hoc queries, such as "show received payments for shipped goods by month by stores". In between runs the ETL (extract – transform – load) process, which updates the data warehouse content.

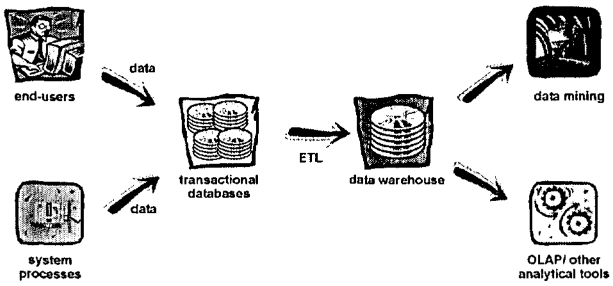

Figure 1: A Modern Information System

Data in the data warehouses, and especially in databases are considered as secure. However, there are many real-life cases where the data are not protected. Furthermore, even if they are protected in the "safe-house" environment of the databases and/or data warehouses, they sometimes leave the environment.

In the next section we describe the data mining process and explain why data mining poses a possible threat to data security.

2 Data Mining

The concept of data mining (DM) has been used since 1991 [Shapiro, 1991], and defined in 1996 [Fayyad, 1996] as a part of process known as knowledge discovery in databases (KDD) which itself is a "nontrivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data". Data mining constitutes only a subtask in the overall KDD process and refers to the information extraction itself.

Data mining is an area that connects many different fields, such as machine learning, artificial intelligence, statistics, databases / data warehouses, computer graphics, inductive logic programming, and many others. The results come in two basic forms – in descriptions and models. In the former, we try to describe, "what is going on in data" [Berry, 2000]. In the latter, we use currently available data and build models, which capture the rules or characteristics in data. The data mining results come in various forms, such as neural nets, decision trees and rules, association rules, clusters, connections (graphs, such as trees and networks) and various scores in form of numbers. There are countless techniques available, such as various methods of decision trees/rules induction, neural networks, genetic algorithms, clustering, statistical models, visualization, and many others. The list of possible results and of available techniques is far from being complete; the purpose is to inform the reader that data mining is not a single tool with a single result; it is rather a process.

As discussed in [Brumen, 2001], data mining is not reserved for large companies (with large databases). In the last few years it has become clear that smaller data sets can be mined too. The problem with smaller companies is that they do not posses in-house expertise for doing data mining, but they do have domain knowledge and understand their data structures much better. They have two choices: not doing data mining at all or doing it with help from outside. The former is sometimes not an option due to competitive reasons; the latter poses a potential security threat to data. The larger companies may require some help from outside as well.

Sometimes the company's resources (hardware and software) may not be adequate for mining purposes, thus the data need to leave the organization. Once outside the safe-house environment of organization's databases and data warehouses, they may be used for purposes other than specified. Due to requirements of most of today's mining tools, data need to be prepared in a flat file. The reason is that many mining tools are developed to avoid DBMS overhead, and to assure compatibility across DB and OS platforms. As such, data are much more easily copied and distributed. Even if we somehow enable access to our DB/DW from outside, we have again no control over what happens once the data are out.

3 Protecting data for outsourced data mining

One approach to protect our data is to use techniques developed for statistical databases. 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 mining world, we can not have data that have been distorted or changed. For example, a decision rule on generalized and not disclosed data would read `IF marital_status = 'not released' ∧ sex='not released' ∧ AGE='young' ∧ zip_code='9xxxx' THEN class = 7`. Obviously, managers would find such form of discovered knowledge useless.

We propose an approach where no data semantics is lost, the statistics inside the data remains intact, but the data are still protected. In our framework, we transform the (relational) database that is to be exported to the outside world. The transformations are to be performed on both data structure and data values.

In the next paragraph we briefly present the notation and basic definitions for relational data model. We adopt the terminology for the relational data model (RDM) from [Elmasri, 1999] and for the abstract data type (ADT) from [Helman, 1994]. The ADT search table operators are not described in this paper; for formal definitions, which are beyond the scope of this paper, refer to [Helman, 1994].

Suppose we have a ***relational database DB*** [Elmasri, 1999], which is a finite set of ***relational schemas*** *DB* = $\{R_1,...,R_i,...,R_l\}$, where R_i is a relational schema, denoted by $R_i(A_1,...,A_n,...,A_N)$, where $A_1,...,A_n,...,A_N$ is a list of attributes. Each attribute A_n is the name of a role played by some domain D_n in the relation schema R_i. D_n is called the domain of A_n and is denoted by $D_n = dom(A_n)$. A relation r_i of the relation schema R_i (also referred to as a search table), denoted by $r_i(R_i)$, is a set of N-tuples, $r_i=\{t_1,...,t_m,...,t_M\}$. Each N-tuple t_m is an ordered list of *N* values, $t_m=\langle v_{m1},...,v_{mn},...,v_{mN}\rangle$, where each value v_n is an element of $dom(A_n)$, or is a special null value. The nth value of tuple t_m, which corresponds to the attribute A_n, is referred to as $v_{mn}=t_m[A_n]$.

A relation $r_i(R_i)$ is a mathematical relation of degree *n* on the domains dom(A_n), which is a subset of the Cartesian product of the domains that define R_i: $r_i(R_i) \subseteq (dom(A_1)\times...\times dom(A_N)\times...\times dom(A_N))$

In the following four definitions we set up a formal framework for reversible database transformation. In lemma 2 we claim that the process is indeed reversible and prove it in proof 2.

Definition 1: Let D_n and D_n^* be sets. A *function* from D_n into D_n^* is a subset *F* of $D_n \times D_n^*$ such that for each element $a \in D_n$ there is exactly one element $b \in D_n^*$ such that $(a,b) \in F$.

Definition 2: Let D^* be a set of domains, such that $D^* = \{D_n^* \mid |D_n^*| = |D_n|\}$. Let $D_n \to D_n^*$ be a function, and $F^D = \{f_n()\}$ be a set of transformations f_n. f_n is said to transform D_n onto D_n^*, if

$1. \forall b \exists a (f_n(a) = b)$

$2. f_n(a_1) = f_n(a_2) \Leftrightarrow a_1 = a_2$

Definition 3 (*a database schema*). Let DB^* be a set of new relations R_i^*, $DB^* = \{R_1^*,...,R_i^*,...,R_L^*\}$, where each R_i^* is denoted as $R_i^*(A_1^*,...,A_n^*,...,A_N^*)$. Each attribute A_n^* is a role played by some domain D_n^* in the relation schema R_i^*. The relation r_i^* of the relation schema R_i^* is a set of N-tuples, $r_i^* = \{t_1^*,...,t_m^*,...,t_N^*\}$. Each N-tuple t_m^* is an ordered list of *N* values, $t_m^* = \langle v_{m1}^*,...,v_{mn}^*,...,v_{mN}^*\rangle$, where each value v_n^* is an element of $dom(A_n^*)$.

The relation schemas of database *DB** are essentially the same as of those in the *DB*. That is, the number of the relation schemas is the same and each schema has the same number of attributes as the corresponding table in schema *DB*. Such a database is needed so that the instances are easily transformed from database *DB* into database *DB**. The transformation function is defined in the next definition.

Definition 4: (*table transformation*). Let us define a function *Trans* that transforms a table instance (a relation), r_i, into a transformed table instance r_i^*, such that:

$$Trans(r_i) = r_i^* = \left\{ \begin{matrix} t_1^*,...,t_m^*,...,t_M^* \mid t_m^* = \langle v_{m1}^*,...,v_{mn}^*,...,v_{mN}^*\rangle, \\ v_{mn}^* = f_n(t_m[A_n]), t_m[A_n] \in t_m, t_m \in r_i \end{matrix} \right\}$$

Lemma 1: Function *Trans* is bijection.

Proof 1: Function *Trans* is bijective if it is injective and surjective.

A function *Trans* is said to be injective, if and only if whenever $Trans(p)=Trans(q) \Rightarrow p=q$. Suppose we have two tuples $p,q \in r_i$, $p=\langle p_1,...,p_N\rangle$, $q=\langle q_1,...,q_N\rangle$. Since $Trans(\{p\})=Trans(\{q\}) \Rightarrow$ $\{\langle p_1^*,...,p_n^*,...,p_N^*\rangle \mid p_n^* = f_n(p[A_n])\} =$ $\{\langle q_1^*,...,q_n^*,...,q_N^*\rangle \mid q_n^* = f_n(q[A_n])\} \Rightarrow$ $p_n^* = q_n^* for\ 1 \le n \le N$. Since by Definition 2 $f_n(a_i) = f_n(b_i)$ when $a_i = b_i$ for $\forall i \Rightarrow$ $p_n = q_n \Rightarrow \langle p_1,...,p_N\rangle = \langle q_1,...,q_N\rangle \Rightarrow p=q$.

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

Suppose we have $q^* \in r_i^*$, $q^* = \langle q_1^*, ..., q_n^*, ..., q_N^* \rangle$. Each q_n^* is calculated as $q_n^* = f_n(p[A_n])$. From definition 2 we have that for function f_n, for each b exists an a such that $f_n(a) = b$. By declaring $s_n^* = b$ it is evident that there must exist a p, so that $Trans(\{p\}) = \{q^*\}$ $\Rightarrow \forall q^* : \exists p(Trans(\{p\}) = \{q^*\})$.▪

Lemma 2: Function *Trans* has an inverse, $Trans^{-1}$, such that $Trans^{-1}(r_i^*) = r_i$, where $Trans(r_i) = r_i^*$.

Proof 2: Let *Trans* be bijection. Then exists one and only one bijection $Trans^{-1}$ that implies

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

That unique bijection $Trans^{-1}$ is called the inverse function of *Trans.*

We first prove that $Trans^{-1}(Trans(\{p\})) = \{p\}$ *for* $\forall p \in r$.
Since *Trans* is injection: $Trans(\{p_1\}) = Trans(\{p_2\}) \Rightarrow \{p_1\} = \{p_2\}$;
Since $Trans^{-1}$ is a function:
$Trans(\{p_1\}) = Trans(\{p_2\}) \Rightarrow$
$Trans^{-1}(Trans(\{p_1\})) = Trans^{-1}(Trans(\{p_2\}))$
i.e. $\{p_1\}=\{p_2\}$.

Next we prove that $Trans(Trans^{-1}(\{q\})) = \{q\}$ *for* $\forall q \in r_i^*$.
Since *Trans* is surjection:
for any $q \in r_i^*$ there exists $p \in r_i$ such that $Trans(\{p\}) = \{q\}$ and since $Trans^{-1}$ is a function: $\{p\} = Trans^{-1}(\{q\})$ is defined for every $q \in r_i^*$.

Thus, $Trans(\{p\}) = Trans(Trans^{-1}(\{q\})) = \{q\}$. ▪

Instead of giving out the original database *DB*, we give out the transformed database *DB**. The set of transformations on domain *D*, F^D, the old names of relations, *R*, and the old names of attributes for each relation are kept secret.

This way the attack on database *DB** is much more difficult since the attacker has no semantic information on the content of the database. If F^D 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 [Adam, 1989], [Willenborg, 1996]. The advantage is that the F^D can easily be chosen so that it corresponds to the value of data to be protected.

For clarity let us take a closer look at an example where we have three tables. We transform them using the above definitions.

Example 1:
Suppose we have a set of domains *D*:
`D={Integer, String, Char},`

a set of relational schemas $DB=\{R_i\}$:
`DB={STUDENT, COURSE, GRADE},`

where each relation schema is denoted as
```
STUDENT(S_ID, Fname, Lname, Zip, City, Age);
COURSE(C_ID, Name, Credits);
GRADE(STU_ID, COU_ID, Grade_Value);
```

and each attribute has the following domain:
```
STUDENT:
Dom(S_ID)={1, 2, 3}, Dom(Fname)=String={John,
Martha, Alice}, Dom(Lname)={Smith, Jones},
Dom(Zip)={12345, 12346, 12347},
Dom(City)={London, Helsinki, Berlin},
Dom(Age)={18, 19, 20};

COURSE:
Dom(C_ID)={1, 2, 3}, Dom(Name)={Math, Physics,
Biology}, Dom(Credits)={5, 10};

GRADE:
Dom(STU_ID)={1, 2, 3}, Dom(COU_ID)={1, 2, 3},
Dom(Grade_value)={A, B, C}.
```

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

student(STUDENT):

S_ID	*Fname*	*Lname*	*Zip*	*City*	*Age*
1	John	Smith	12345	London	18
2	Martha	Smith	12346	Helsinki	19
3	Alice	Jones	12347	Berlin	20

course(COURSE):

C_ID	*Name*	*Credits*
1	Math	5
2	Physics	5
3	Biology	10

grade(GRADE):

STU_ID	*COU_ID*	*Grade_Value*
1	1	A
1	2	B
2	1	A
2	3	B
3	1	C

We define *D** as
`D*={Number1, String, Number2}.`

Further, we define
`DB*={TABLE1, TABLE2, TABLE3},`

where each relation schema is denoted as

```
TABLE1(COL1, COL2, COL3, COL4, COL5, COL6);
TABLE2(COL1, COL2, COL3);
TABLE3(COL1, COL2, COL3);
```

and each attribute has the following domain:

```
TABLE1:
Dom(COL1)={10, 13, 16}, Dom(COL2)={Str1, Str2,
Str3}, Dom(COL3)={Str4, Str5}, Dom(COL4)={37042,
37045, 37048}, Dom(COL5)={Str6, Str7, Str8},
Dom(COL6)={61, 64, 67};

TABLE2:
Dom(COL1)={10, 13, 16}, Dom(COL2)={Str9, Str10,
Str11}, Dom(COL3)={22, 37};

TABLE3:
Dom(COL1)={10, 13, 16}, Dom(COL2)={10, 13, 16},
Dom(COL3)={8, 9, 10}.
```

By applying the function *Trans* on each of the relation instances, `student(STUDENT)`, `course(COURSE)` and `grade(GRADE)`, we get the following transformed instances:

Trans(`student(STUDENT)`*)*=`table1(TABLE1)`
Trans(`course(COURSE)`*)*=`table2(TABLE2)`
Trans(`grade(GRADE)`*)*=`table3(TABLE3)`

The instances are a set of tuples. We present each of the instances in a tabular form:

`table1(TABLE1)`:

Col1	*Col2*	*Col3*	*Col4*	*Col5*	*Col6*
10	Str1	Str4	37042	Str6	61
13	Str2	Str4	37045	Str7	64
16	Str3	Str5	37048	Str8	67

`table2(TABLE2)`:

Col1	*Col2*	*Col3*
10	Str9	22
13	Str10	22
16	Str11	37

`table3(TABLE3)`:

Col1	*Col2*	*Col3*
10	10	10
10	13	9
13	10	10
13	16	9
16	10	8

■

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, and a set of clusters, to name only a few. Basically, the models can be built using supervised or unsupervised learning. In the former, the algorithm takes as input a vector of values and returns the class as the output. The calculated class is in learning phase compared to the actual value and the correction is made if needed, thus supervised learning. In the latter, the algorithm tries to group vectors together based on some similarity function. Since there is no correct answer, there is nothing to supervise.

For example, when building a decision tree, the 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.

Any mining result that is returned is again in form of a model that includes the transformed data elements. Thus, the result that is returned can be taken as a set of relation schemas $DB^*=\{R_i^*\}$ with a corresponding set of relation instances $\{r_i^*\}$, and connections among them, depending on the structure of the model. Note that even a single data cell can be viewed as an instance of a table with one attribute (column) and one tuple (row).

With inverse transformation, the data owner decodes the model (relation instances) into a readable form. The example 2 depicts the process.

<u>*Example 2:*</u>
Suppose the data mining rule says that `IF table1.Col6 < 67 THEN table3.Col3 > 8.`

We have two table instances, q^* (i.e. `table1`) and s^* (i.e. `table3`). Each table instance has only one tuple, i.e. $q^*=\{t_1^*\}$and $s^*=\{u_1^*\}$

`table1`: q^*

Col6
67

`table3`: s^*

Col3
8

First, by using the inverse, $Trans^{-1}$ on q^*, we get:

$Trans^{-1}(q^*)=q=\{t_l\}=$
$=\{t_l | t_l\,\texttt{[Age]}=f_1^{-1}\,(t_1^*\,\texttt{[Col6]})\}=$
$=\{t_l | t_l\,\texttt{[Age]}=(t_1^*\,\texttt{[Col6]-7)/3}\}=$
$=\{t_l | t_l\,\texttt{[Age]=(67-7)/3}\}=$
$=\{t_l | t_l\,\texttt{[Age]=20}\}$

Next, by applying $Trans^{-1}$ on s^*, we get:

$Trans^{-1}(s^*)=s=\{u_l\}=$
$=\{u_l | u_l\texttt{[Grade\_Value]}=f_3^{-1}\,(s_1^*\,\texttt{[Col3]})\}=$
$=\{u_l | u_l\texttt{[Grade\_Value]}=f_3^{-1}\,\texttt{(8)}\}=$
$=\{u_l | u_l\texttt{[Grade\_Value]= C}\}$

Since instances q^* (`table1`) and s^* (`table3`) correspond to instances `student` and `grade`, respectively, we get the following two relation instances as a result:

`student`: q

Age
20

`grade`: s

Grade_Value
C

Finally, the rule decodes to `IF Student.Age < 20 THEN Grade.Grade_Value > C.` •

4 Conclusion

The information age has caused a shift from a product-centric society to a data-centric one. While the fundamentals for data storage (and protection) have been around for almost 40 years, and have become very solid since the introduction of relational database systems, the new technologies, such as data warehousing and data mining, require special attention and care.

In the paper we presented a new knowledge discovery technology – data mining – from the data security perspective. In data mining process, sometimes the data need to be modeled outside of an organization. In this case they need to be protected in such a way that the modeling process is still possible and the data security is preserved at the same time.

We presented a formal framework for transformation of a schema and content of a relational database. We prove that the transformation is reversible. With careful selection of transformation functions, the attack on data becomes infeasible. The functions can be selected so that the effort to break them exceeds the value of protected data. By using the framework the outsourced data miner is still able to do the data mining, and the data remain secure. The reversibility of the process enables the data owners to decode the model into a readable form.

The presented framework is especially useful for models-based data mining, where the data values and data structure play no role when building a model.

References

[Adam, 1989] Adam, Nabil R.; Wortman, John C.: Security-Control Methods for Statistical Databases: A Comparatice Study, ACM Computing Surveys, Vol. 21, No. 4, pp. 515-556, 1989

[Berry, 2000] Berry, Michael A. J.; Linoff, Gordon: Mastering Data Mining: The Art and Science of Customer Relationship Management, John Wiley & Sons, Inc., New York, USA, 2000

[Bigus, 1999] Bigus, Joseph P: Data Mining with Neural Networks: Solving Business Problems from Application Development to Decision Support, McGraw-Hill, New York, USA, 1999

[Brumen, 2001] Brumen, Boštjan; Welzer, Tatjana; Golob, Izidor; Jaakkola, Hannu: Convergence Detection in Classification Task of Knowledge Discovery Process, In Proceedings of Portland international conference on management of engineering and technology, Portland, Oregon, USA, 2001

[Elmasri, 1999] Elmasri, Ramez A.; Navathe, Shamkant B.: Fundamentals of Database Systems, 3rd Edition, Addison-Wesley Publishing, New York, 1999

[Fayyad, 1996] Fayyad, Usama; Shapiro-Piatetsky, Gregory; Smyth, Padhraic; Uthurusamy, Ramasamy; (Eds.): Advances in Knowledge Discovery and Data Mining, AAAI Press, USA, 1996

[Hedberg, 1995] Hedberg, Sara R.: The Data Gold Rush, Byte Magazine, 10-1995

[Helman, 1994] Helman, Paul: The Science of Database Management. Irwin Inc, 1994

[Hundepool, 1996] Hundepool, Anco J.; Willenborg, Leon C.R.J.: μ- and τ-Argus: Software for Statistical Disclosure Control, Proceedings of 3rd International Seminar on Statistical Confidentiality, Bled 1996

[Moore, 1965] Moore, Gordon E.: Cramming More Components Onto Integrated Circuits, Electronics, Vol. 38, No. 8, April 19, 1965

[Reinhardt, 1995] Reinhardt, Andy: Your Next Mainframe, Byte Magazine, 5-1995

[Samarati, 2001] Samarati, Pierangela: Protecting Respondents' Intentities in Microdata Release, IEEE Trans. On Knowledge and Data Engineering, Vol. 13, No. 6, 2001

[Shapiro, 1991] Piatetsky-Shapiro, Gregory; Frawley, William J. (Eds.): Knowledge Discovery in Databases, AAAI/MIT Press, USA, 1991

[Sweeney, 1997] Sweeney, Latanya: Guaranteeing Anonymity when Sharing Medical Data, the Datafly System, Proceedings, Journal of American Medical Informatics Association, Washington, DC, Hanley & Belfus Inc., 1997

[Willenborg, 1996] Willenborg, Leon C.R.J.; De Waal, Ton: Statistical Disclosure Control in Practice, LNS Vol. 111, Springer-Verlag, 1996