Inference Attacks and Control on Database Structures

Muhamed Turkanović [1], Tatjana Welzer Družovec [1], Marko Hölbl [1]

[1]*University of Maribor, Faculty of Electrical Engineering and Computer Science, 2000 Maribor, Slovenia*

***Abstract* – Today's databases store information with sensitivity levels that range from public to highly sensitive, hence ensuring confidentiality can be highly important, but also requires costly control. This paper focuses on the inference problem on different database structures. It presents possible treats on privacy with relation to the inference, and control methods for mitigating these treats. The paper shows that using only access control, without any inference control is inadequate, since these models are unable to protect against indirect data access. Furthermore, it covers new inference problems which rise from the dimensions of new technologies like XML, semantics, etc.**

***Keywords* – Inference, Attacks, Database, Security, Semantics.**

1. Introduction

In our information-based society, databases present the foundation for most aspects of our modern life. We can encounter with them directly or indirectly, aware or unaware, and willingly or unwillingly. We can also be personally involved, e.g. visiting a doctor, borrowing a book from a library, singing into a social network, etc., or complementarily, e.g. buying groceries or cinema tickets, visiting a web page, etc. When personally involved, some of our personal information (e.g. name, address, social number, etc.) is being stored in databases we do not manage and do not have direct control over it. However, due to security policies regulated by different government laws, personal information of users or clients has to be kept secret and secured [1]. Using various Database Management Systems (DBMS) organizations have the ability to provide some level of protection and security for the stored data. DBMS mainly engage with access control in order to secure data, thus prevent unauthorized access from unprivileged entities. Privacy concerns are especially prominent in e-commerce, e-government and e-healthcare. Since such e-services hold highly sensitive data, the stakes for cybercrime are also higher and security features engaged by the DBMS may not be enough to stop a highly motivated adversary. Such an adversary could infer sensitive information beyond privileges from data to which he is granted access.

When a user is able to infer sensitive information to which he/she is not granted access, by using authorized query results and prevailing common knowledge, this is called an inference attack [2]. The inference problem was firstly noticed and studied in statistical databases, where aggregated data of a group disclosed information about a particular individual [3]. In 1980s the study on inference shifted to relational databases and since then the inference problem emerged on various types and forms of data and databases, i.e. multi-level secure database systems, outsourced encrypted databases, semi-structured databases, object-oriented databases, XML databases, multimedia database management systems, social networks,geolocated data, etc. [3-10].

A general solution for the inference problem is highly difficult to achieve, due to various different inference channels which can be applied on the aforementioned types and forms of data and databases. However multiple domain-specific solutions were already presented and used in practice.

The remainder of the paper is organized as follows. Section 2 provides some brief preliminaries on database security. The inference problem in detail is discussed in Section 3. Some inference control techniques are presented in Section 4. Section 5 presents some emerging inference problems, and finally we conclude the paper in Section 6.

2. Database Security

Databases became highly robust and refined structures through multiple decades of evolvement. Databases like all other evolving structures gained high interest and thus related attention by malicious users who want to exploit their weaknesses and imperfections for their personal goals. The security of databases is not negligible and is still a growing concern considering the number of incidents reported constantly (e.g. Sony, USA Government, Mc'Donalds, etc.) [11-13]. In consideration of the stakes involved (e.g. government laws protecting privacy and fines if failing to comply), the security of databases is thus highly important [14, 15].

Database security is based on availability, integrity and confidentiality, whereby availability deals with the avoidance of hardware and software

errors and providing access to information whenever needed; integrity with the protection from unauthorized data access and illegal modification; and confidentiality with the protection of unauthorized data disclosure [16]. Typical layers of database security can be classified into physical security, network security, encryption, authentication, auditing, and backups [1, 16]. The physical security depends on the security of the host machine and the operation system. The network security layer engages with firewalls and network-based intrusion detection systems. Traditionally the main focus is on the authentication layer i.e. access control, managing user privileges to database objects. There are three access control models which prevent direct unauthorized access to data i.e., Discretionary Access Control (DAC), Mandatory Access Control (MAC), and Role-based Access Control (RBAC) [17]. If not properly configured DAC can be exploited by a malicious user by giving access right to a user from lower security level. Since MAC is controlled from a single administrator, such problems are avoided.

Privacy intrusions take place even in the presence of privacy and access control systems due to social engineering and inference channels.

3. The Inference problem in databases

This section briefly describes the inference problem while presenting some basic preliminaries and examples. In addition the section presents additional information on inference problem in different forms.

Categorizing database inference to a group of information security attacks is very hard, due to the fact that it leverages the human mind and a logic approach in order to infer secure data. For better understanding of the topic we firstly give the meaning of the word “inference”, which is “forming a conclusion from premises” [18]. Inference occurs when a malicious user infers some protected or private data without directly accessing it [19]. In this example “direct access” denotes an access or a query which is allowed for a specific database user. For example inference can also occur on Discretionary Access Control (DAC) systems where users have rights and grants to explicitly access only parts of the database. As mentioned in the previous section, database protection methods like access control only protect data from direct attacks, thereby leaving a gap for highly motivated and skilled adversaries who use indirect inference channels to access secure data. Such adversaries can be hackers, paid by a private company to infer secured data of a rival company; to infer private information on an employee, etc. Other examples are e-healthcare services and medical databases which are especially vulnerable to such attacks because they store private and confidential information in conjunction with corresponding medical domain knowledge.

An example of an inference problem in healthcare was presented by Farkas et al. [3], where a scenario of a fictive Jane Smith was discussed. Consider Jane Smith who undergoes some genetic tests to determine if she has mutated BRCA1 and 2 genes, which indicate increased risks for developing breast cancer. Ms. Smith pays for the testing herself and instructs her physician not to release any information about the testing to her health insurance company. Unfortunately the test results were positive and indicate a high possibility for breast cancer. Ms. Smith’s physician immediately schedules her for mammography every six months in order to keep an eye on a possible breast cancer. Ms. Smith regularly goes to the mammography, and the bills for it are sent to her insurance company. Since regular mammography is done only on possible breast cancer patients Ms. Smith’s insurance company terminates her insurance and thus violates her privacy without consequences.

Releasing authorized data can also lead to indirect disclosure of other related but private data via inferences. An example of such an inference problem can be seen in an extensive data re-identification experiment run in 1990 by the United States Government [16]. Data re-identification is a method for generating anonymous data in data warehouses through static manipulation of stored sensitive data. The goal of the experiment was to determine how many individuals were identifiable only by some non-sensitive demographic values. The results showed that 87% of 248 million US citizens could be uniquely identified based on the combination of gender, date of birth and a five-digit ZIP code.

The mere existence of various inference vulnerabilities are because of the interconnections between non-protected, non-sensitive and accessible data provided to the users, and the protected sensitive data [2]. Such an interconnection is unavoidable and very hard to detect.

3.1. The AERIE model

A characterization of the inference problem was presented as a model called AERIE (Activities-Entity-Relationships-Inference-Effects) by a research project at the University of Alabama [20]. It engages with the entity-relationship model (ER-model), whereby it characterizes possible inference channels in terms of an entity, activity, and relationships. The model correlates desired secret data with the target of an inference attack, which in turn is expressed in

terms of a construct of the model. According to the model, there are six types of a target: Entity Materialization, Activity Materialization, Entity-Entity Relationship, Activity-Activity Relationship, Entity-Activity Relationship, and Relationship-Relationships Relationship.

An example of an Entity Materialization inference would be infering that a growing season (i.e. entity) in a farming community is underway based on the increased database entries of sold fertilizers, seeds or pesticides. An Activity Materialization example is when based on ordering of an ice axe and low-temperature sleeping bags, one can infer that a winter mountain climbing expedition (i.e. activity) is about to occur. An Entity-Entity Relationship inference example is when one can infer a company (i.e. entity) that is supporting a project (i.e. entity) based on an employee of the particular company attending a pro-project meeting. An Activity-Activity Relationship inference example is when one can infer that cotton picking (i.e. activity) has occurred based on daily database entries of ginning (i.e. activity) activities. An example of the Entity-Activity Relationship is when one can infer that a company (i.e. entity) is adopting new manufacturing processes based on orderings (i.e. activity) of new equipment. For the Relationship-Relationship Relationship consider two relationships. One is a sorted list of student names. Another is a classroom setting based on the student names and numbers. One could infer that grades associate with each student's name.

The examples of the AERIE model presents the fact that database owners should always assume that an adversary has extensive familiarity with the specific domain of knowledge related to our database entities and is going to use it to indirectly access secure and private data.

3.2. Inference Rules

In order for a database administrator or a database security developer, to understand the inference problem, he/she needs to understand the possible inference attacks. In the late 1980s, Yip and Levitt identified six inference rules, which an adversary could use to infer data from a secure and protected database [16, 19]. The rules cover the intersection, difference and union relationships between multiple query results [19]. The rules were used to prepare an inference detection system, which tries to detect if an adversary could indirectly access data using combinations of multiple queries. The identified inference rules are: split queries, subsume queries, unique characteristics inference, overlapping inference, complementary inference, and functional dependency inference.

When a query can be split into two smaller inferred queries, which are in respect to one another, this is called split queries. The idea is to identify some returned records of a query which are related to some other query and use it to deduce or infer some information which was meant to be protected. For example, an adversary can issue a query requesting salaries of all employees aged 40. The query returns two records. The adversary then issues a query requesting the age of all managers, which returns one record of 40. The adversary can now conclude that the bigger value of the first query is the salary of the manager.

The subsume queries rule searches for possible relation between two query results, whereby the first query result returned records which all share a common attribute. This common state could be used by an adversary to infer data from the second query. For example, an adversary issues a query requesting the salary of all employees aged 30. The query returns one record, meaning that all employees aged 30, have the same salary. Since the adversary knows, that a specific employee is 30 years old, he/she infers that his salary is the one from the first query.

When a certain group of attribute values of a query result uniquely identifies a record, or when all but one record of a query result could be obtained by another query, this is called the unique characteristics rule. For example, let us assume that an adversary knows that only one manager in a company is 40 years old. He/She than runs two different queries requesting the attribute pair AGE-SALARY. The first query request all employees older than 40 and the second query, all non-manager employees older than 40. If the query results are the same except of one record than we know the age and salary of the manager.

The overlapping rule tries to identify commons of multiple queries in order to obtain sensitive information. For example an adversary issues multiple queries requesting the salaries of all the janitors, the salary of all the employees aged 40, and the salary of all the employees from the town Highville. The adversary than uses the intersection on those query results, and if it returns one record, he inferred the age, town and a salary of a specific janitor.

In contrary to the overlapping, the complementary inference uses the difference between multiple query results, i.e. negating the overlapping inference, while functional dependency inference leverages the database schema level dependencies [16].

As stated by Yip and Levitt, these rules are sound but they are not complete, since inference attacks can be highly unpredictable and relying only on such rules is insufficient [19]. The rules are a

good start when trying to identify the basic inference vulnerabilities of a database, since they can present possible inference channels through the correspondence of multiple query results.

3.3. Data Mining

In the logical inference type of attacks an adversary uses association rules to make logical assumptions about data. Such association rules are like those apparent from data mining processes [21]. Data mining techniques are used for searching patterns and building rules from data and associations of those. Adversaries usually know what they are looking for, while engaging with an inference attack attempt. An adversary with high pre-knowledge of a domain which is part of his desired but secure data is likely to be much more efficient in an inference attack. For such preliminaries adversaries could turn to data mining processes which can utilize any form of publicly available collected data in order to gain additional pre-knowledge. The correlation with inference and data mining can be seen in the example that data mining techniques could predict associations or induce multiple records of a particular word inside of a group of documents stored in a database [21]. Having some predictions, they could be put into rules and applied in inference techniques to infer the desired data.

3.4. Statistical Databases

The inference problem was firstly studied in statistical databases and mentioned in literature in 1970s with contributions of Dalenius and Schlörer, whereby Schlörer presented the database community and Dalenius the statistical community [2, 22]. The interest in the statistical community shifted to inference control in the 1990s where the discipline was further developed under the names of Statistical Disclosure Control and Statistical Disclosure Limitations [22].

Statistical databases are used to provide simple summary statistics on data sample groups stored in databases. The goal of statistical databases is to cumulate new information based on the stored data without exposing single individual values. The group samples in statistical databases are mostly groups of individuals, i.e. population, and are used for e-commerce, medical research, census taking, etc. Since a single sample of such a group model is an individual, exposing data stored with it would mean exposing private and personal information. Because of such operating environment, statistical databases have to be highly secure and engaged in special security features. In the past released information was mainly in the form of microdata, i.e. pre-computed statistics, while in contrary today the demand is to release the specific data, i.e. microdata, to allow additional analysis on them when needed.

Providing simple summary statistics in statistical databases means using only database query functions like SUM, COUNT, AVERAGE, etc. [23]. Hence, any query which does not use such functions is not allowed or rejected (e.g. SELECT * FROM table). Providing such a security environment is relatively simple but the real security problem of statistical databases is how to allow such simple summary statistics on protected and information while preventing any form of compromise like inference.

A statistical query is more formally representable in the form SF(QE, N), where SF stands for Statistical Function, N is an attribute (e.g. salary) to which this function is applied, and QE is a qualification expression which can have multiple query sets (QS) connected by logical operators (i.e. AND and OR) [24]. An example of a statistical query is: SUM((GENDER=Male AND D-BIRTH=1987), SALARY). In this example the query expression [(GENDER=Male AND D-BIRTH=1987)] has two query sets (GENDER=Male) and (D-BIRTH=1987), connected with an operator AND. The attribute on which the SF is applied is SALARY. The explanation of the query is “Looking for the sum of the SALARY for all individuals who are male, and were born in the year 1987”. The complexity of a query is proportional to the number of queried attributes in the query expression, meaning that more attributes are in the query expression, more likely it is that confidential information is disclosed [24].

3.5. Example and attacks on statistical databases

For the better perspective, we show an example of an inference attack on the statistical database. Let us presume that an adversary has access to a database of some governmental healthcare project. As aforementioned, he/she can only use statistical functions to aggregated data and reveal some statistical information. At the beginning he/she constructs a query:

[1] SELECT COUNT WHERE Gender=Female AND ZIP_Code=19000 AND Sterilization=Yes.

Let us assume that query 1 returns the value 115. The adversary now knows that there are 115 women from the town with the ZIP code 19000 (e.g. let us assume this is a fiction town with name Highville) which have undergone a sterilization procedure. Since this is mainly a statistical value it is totally legitimate and could be used for further statistical evaluation. The

adversary now goes further and constructs another query by adding additional query sets:

[2] SELECT COUNT WHERE Gender=Female AND ZIP_Code=19000 AND Sterilization=Yes AND Y-Birth=1978.

Now, suppose query 2 returned the value 15. The adversary added one additional query set (Y-Birth=1978) which further aggregated the data and revealed that there are only 15 women in Highville which were born in the year 1978, and have undergone the sterilization procedure. The query result and the query itself are still legitimate after the modification since no individual and personal information were revealed, thus there are no privacy violations. As aforementioned an adversary usually knows what he/she is looking for while executing inference attacks, meaning he/she alters the queries in such a way that they reveal the most targeted results. We also mentioned that an adversary has usually some pre-knowledge of the data domain, meaning he/she could know about some subgroups which could be accumulated inside the database, thus use such knowledge for better inference attacks. Having this in mind let us assume the adversary is a governmental employee who suspects his/her coworker has undergone a sterilization procedure. He/She knows that his/her coworker lives in Highville and was born in the 1978. Till know, without violating any privacy policy, the adversary found 15 women with such a profile who have undergone the sterilization procedure. One of these women could be his/her coworkers. Using his/her domain and pre-knowledge the adversary further alters the query by adding an additional query set:

[3] SELECT COUNT WHERE Gender=Female AND ZIP_Code=19000 AND Sterilization=Yes AND Y-Birth=1978 AND Governmental_Employee=Yes.

Unfortunately, query 3 returned the value 1. The adversary has now found only one person which accommodated with the query set search conditions. Having this indirect information the adversary can be almost sure that the query result is pointing to his/her coworker, thus revealing confidential information without breaking the security policy of the database. This type of an inference attack is called Single Match [25].

To mitigate the inference problem, a highly simple method as a solution comes to foreground, i.e. monitoring each query and disabling those aggregated query results which include only one tuple of data, hence avoiding the problem of revealing confidential information of a single record. Moreover a threshold could be set higher (e.g. 5), thus further lowering the probability of inducing confidential and secret information of a single record. Using such a technique, our adversary would be deprived of the last query result since it included only a single record of aggregation, thus disabling him/her the possibility of revealing confidential information of a possible coworker. This method is called Query Set Size Control (QSSC) [26]. Unfortunately, as much as this would be a solution for the aforementioned problem, this is only a fraction of the inference possible and would not cover all the inference security. For demonstration, an adversary could use multiple query results with different query sets which are over the threshold, and run multiple complements on those results, thus still being able to reveal confidential information. Such an approach leads to attacks called trackers [26] and QSSC is ineffective against such attacks. Since solving the inference problem is highly complex, we will discuss the Inference Control and its methods in a separate section.

Other inference attacks on statistical databases are Arithmetic Means, Diophantine inference or linear system attack, Addition aggregate, and Partitioning [25]. The aforementioned trackers are an example of the Diophantine linear system (DLS) attacks. The DLS attack leverages the possibility of having several query results with different sets of records. Using such gathered data an adversary can construct a linear system and try to solve it in order to obtain unknown data or confidential information. To solve the linear systems the adversary constructs several Diophantine's linear equations based on values already gathered. The Diophantine equations are indeterminate polynomial equations and are used for finding integers, whereby there are always fewer equations than unknown variables [27]. An example of the equation is:

$$A\,x1 + B\,x2 + C\,x3\ldots = R1$$

Here R1 represents the size of the records and x1, x2, x3 the unknown variables. Since Diophantine equations define algebraic curves, the linear system is constructed as a matrix of equations and solved with the help of algebra:

$$A\,x1 + B\,x2 + C\,x3\ldots = R1$$
$$A\,x1 + B\,x2 + C\,x3\ldots = R2$$
$$\ldots$$
$$A\,x1 + B\,x2 + C\,x3\ldots = R3$$

While solving the system, the individual unknown variables and thus results are revealed.

Further, the Arithmetic attack leverages the arithmetic means of a query set whose cardinality is

in the range [a, N-a], where N*2*a is the total number of records in the database [25]. Explained, such attack can reveal the size of a table by having results of the averages of varying fields of the same table. With such gathered knowledge an inference derivation can easily be started. In addition aggregate attack leverages the SUM function to infer protected information. By using multiple SUM query results an adversary could disclose information by calculating the difference between those. The partitioning attack in contrary uses low-frequency groups to infer new information [25]. Low frequency groups consist of a small number of entities which in contrary make a large proportion of data. The idea is that an adversary leverages multiple queries which return a small number of data and uses them to cancel each other out, leaving the desired data.

As it can be seen, the problem with inference is that there were no direct security violations, since such would be detected and blocked by DAC or MAC. The real issues are the indirect, highly logical attacks which work under the rules of the security policy and are thus undetected. Since inference attacks cannot be foreseen, database security managers always have to assume that an adversary is highly motivated and has a broad spectrum of domain knowledge.

Since multiple types of statistical database exist, some are more vulnerable to inference attacks than others [25]. Of the eight types of statistical databases, i.e. delay, dynamic, static, dedicated, centralized, decentralized, immediate and shared, the last three have the biggest risks of inference attacks.

4. Inference control and data anonymity

A solution for the inference problem, which would be general, is highly difficult or even impossible to achieve, since an adversary could apply some deep pre-knowledge in performing an inference attack [20]. The inference problem is specific in a way that it is almost an utopian goal, hence it cannot be solved by traditional access control methods, as the disclosure of sensitive information is not derived from unauthorized access but from authorized ones which use high developed background knowledge of a specific domain [2]. This section presents some most prominent inference control methods and techniques and helps understand the complexity of the inference control.

4.1. K-Anonymity

Having in mind the above aforementioned experiment in section 3, where 87% of included US citizens could be uniquely identified based on the combination of gender, date of birth and the ZIP code, a question comes to mind, how to publicly release a database without compromising individual privacy? Based on the experiment's result it can be seen that hiding unique identifiers, like name or social security number is not enough, since data released could be linked with other publicly available information and used to re-identify an individual.

A model for protecting privacy called k-anonymity was introduced in the late 1990s by Samarati and Sweeney [28]. It enables sensitive information to be released without threatening privacy. The objective for k-anonymity models is to disseminate statistical microdata via generalization and suppression in a way that individual privacy is sufficiently protected against recognition of the subjects to which it refers.

K-anonymity model tries to satisfy one of the main requirements of the statistical community to release data which should be indistinguishably related to less than a certain number of records, i.e. every tuple released cannot be related to fewer than k respondents, i.e. the sensitive information of the tuple should be identical with at least k-other tuples [16, 29]. It is important to mention that all requirements of k-anonymity apply to release tables only, and not on the data available externally to adversaries which can be used for linking. A formal definition of k-anonymity requirement is as follows [29]:

Definition 4.1. Each release of data must be such that every combination of values of quasi-identifiers can be indistinctly matched to at least k respondents.

Quasi-identifiers represent sets of attributes included in the private table, which are also externally available and can be exploited for linking. These are non-explicit identifiers which remain after explicit identifier (e.g. name or social security number SSN) are removed.

K-anonymity is mainly used as a static method for increasing data anonymity and is as such not directly an inference control method but uses some techniques which are similar or the same those of inference control. Furthermore, by assuring k-anonymity on released data, some obvious inference channels are closed. As already mentioned, k-anonymity focuses on two techniques in particular to increase data availability i.e. generalization and suppression [16]. These techniques are highly desirable since they preserve the quality of individual tuples, i.e. truthfulness of the information, as in contrary to scrambling, sampling or swapping. Generalization uses general values to substitute them with some value of a given attribute (e.g. postal address generalized to a street, dropping the number). Suppression is used as a method which can reduce the amount of generalization necessary to satisfy the

k-anonymity constraint. In other words, suppression is used to moderate the generalization process when a limited number of tuples with less than k occurrences would force a great number of generalizations [29].

The main difference between typical inference control and the k-anonymity process is that inference control typically hides sensitive values through aggregation or by blocking the retrieval of query results, while k-anonymity releases the whole package including the sensitive values, but anonymously [26]. Thus each record in a table under k-anonymity requirements, shares the same identifying values with k-1 other records, hence any attempt of linking a record with an individual i.e. real person, ends up with at least k indistinguishable choices.

For the better understanding we show an example of k-anonymity using the same scenario as in section 3. Let us assume we have a database populated with

Table 1.Private table with medical and personal data.

Name	SSN	Date of birth	Sex	ZIP	Governmental employee	Procedure	…
Jackson	12502	1978	F	19000	Y	Sterilization	…
Smith	85462	1987	F	19120	Y	Kidney Transplant	…
Li	21511	1982	F	19888	N	Sterilization	…
Rajesh	51512	1978	M	19120	Y	Hearth Transplant	…
Manderly	22151	1965	F	19000	Y	Sterilization	…
McMillen	56222	1959	F	19000	N	Sterilization	…
…	…	…	…	…	…	…	…

Table 2.K-anonymity private table.

Name	SSN	Date of birth	Sex	ZIP	Governmental employee	Procedure	…
Person	Number	78-90	F	19***	Not_released	Sterilization	…
Person	Number	78-90	F	19***	Not_released	Transplant	…
Person	Number	78-90	F	19***	Not_released	Sterilization	…
Person	Number	78-90	M	19***	Not_released	Transplant	…
Person	Number	55-76	F	19***	Not_released	Sterilization	…
Person	Number	55-76	F	19***	Not_released	Sterilization	…
…	…	…	…	…	…	…	…

data from a governmental health project. In this example the government cannot initialize a statistical database since macrodata, i.e. aggregated data is not enough for a specific research. The research group funded by the government requires the microdata in order to fully engage with the analysis, but is still not allowed to see private and sensitive information of individuals. The governmental group responsible for the security of the gathered data has to release the database (e.g. Table 1) without compromising individual privacy.

The group responsible for the security of the data decides to use the k-anonymity concept to ensure data anonymity before releasing the private and secure data. The group engages with the generalization technique described earlier to increase the anonymity of the data. The result of the generalization process can be seen in Table 2.

Since the new table satisfies the k-anonymity requirement, it can be released without the fear of exposing any individual's private information.

4.2. Inference control subdisciplines

The inference control has evolved into three sub-disciplines, i.e. Tabular Data Protection (TDP), Queryable Databases (QD), and Microdata Protection (MP) [22]. We further briefly describe each of the subdisciplines separately.

4.2.1. Microdata protection

The MP is the youngest subdiscipline and is continuously evolving. One of the emerging concepts of microdata protection is k-anonymity, which we presented in the previous subsection [30]. As already defined earlier in the previous subsection, the goal of Microdata protection is to generate such microdata (e.g. Table 2), which defers from the original set of data (e.g. Table 1) in such a way that the disclosure risk is low, meaning that the risk of an adversary using the generated and protected data (e.g. Table 2) to determine confidential information on a specific individual among those in the original set (e.g. Table 1) is highly low. Another requirement of microdata protection is that user analysis on original data set (e.g. Table 1) and protected data set (e.g. Table 2)

yields the same or at least similar results [22]. Methods of microdata protection, regardless to k-anonymity concept, can be classified into two categories, i.e. masking original data and generating synthetic data. The first category consists of perturbative and non-perturbative methods, whereby both aim to generate modified versions of the original microdata sets. Before releasing the data the first category distorts the data using noise addition, aggregation, swapping, etc., while the second does not alter data, but uses suppression or reduction of detail [22]. The second category, i.e. generating synthetic data, in contrary tries to generate data while preserving some statistical properties of the original one.

4.2.2. Tabular Data Protection

TDP is the oldest and best established subdiscipline of the inference control and was used by governmental statistical agencies worldwide to publish static aggregated information [22]. Such public releases had to be backed up with a security aspect in such a way that no confidential information of an individual among those inside the aggregated data could be inferred. For tabular data, information could be extracted by small cell counts in categorical data, e.g. identifying a small hospital in a particular region based on the small number of reported admissions [31]. Other conceivable indirect attacks are: external attack, internal attack, and dominance attack [22]. An example of an external attack is when there is a frequency table "JOB – TOWN" released, and there is only one individual corresponding to a particular Job or a Town. Furthermore a magnitude table with the average salary for each Job or Town is also released, thus exposing the exact salary of the only individual with a particular Job or Town. In case when there are only two individuals corresponding to a particular Job or Town, an internal attack can take place, since the salary of each of those two individuals is disclosed to each other. At last, the dominance attack is when an individual dominates with a contribution of an attribute, i.e. cell of magnitude table, with such scale that his/her contribution upper bounds the contributions of the rest. For example if an individual's salary covers 90% of the sum of all salaries for a particular Job, the same individual has inferred that his co-workers are not doing financially well.

As with the MDP, the TDP has also two groups of methods for disclosure control, i.e. non-perturbative and perturbative methods. The best known non-perturbative method is the Cell-Suppression method (CS), whereby cells which are identified as sensitive by the so-called sensitive rule, are simply suppressed. The sensitive rules are the dominance rule and the p%-rule [22]. The perturbative methods output tables with modified values. Examples of such methods are Controlled Rounding and Controlled Tabular Adjustment. The first rounds values in tables to a multiple of a rounding base. The second causes less information loss since it tries to find such a table which is highly similar to the original one, but still ensures a protection for all sensitive cells.

The importance of TDP demonstrates the fact that The National Center for Health Statistics of The United States sponsored the development of a software for TDP or precisely disclosure limitation software for two-dimensional tables which was developed by OptTek Systems, Inc. [31].

While choosing an inference protection technique it is also important to keep the technique chosen as secret. This is due to the fact, that an adversary knowing which technique was used to create a protected data set, could use this information to start technique-specific re-identification attacks in order to disclose confidential information. Unfortunately, keeping the technique used secret is not an easy assignment, since an adversary could leverage publicly available database information which gives him/her some clues on the technique used to protect the data. As already mentioned, inference attacks can be unpredictable, since adversaries could be highly motivated and have deep knowledge and skills.

4.2.3. Queryable Data Protection

The idea behind the QD inference control is that aggregated information obtained by a user as a result of multiple successive queries should not enable him to infer specific private and confidential information of an individual. The QD inference control approaches try to protect a confidential vector of numerical data from disclosure through query answers. There are three approaches or methods to prevent such attacks, i.e. perturbation, query restriction (QR), and camouflage [22]. The QD inference control methods differ from each other depending on the correctness of the query results a user requires. If a user does not require deterministically correct answers to queries, the best and simplest method for such is Data Perturbation. The perturbation methods can be applied as input, i.e. before a query, or output, i.e. after a query. The input perturbation usually clones the original data and generates a perturbed database for the user to access. The user can then issue different queries which are not spied. This type is highly bias because of the noise added and is not suitable for dynamic databases [2]. In contrary the output perturbation approach does not change the database, i.e. it does not add noise, rather deals with query replies in a way the results

are rounded down or up. This approach does not have any bias problems as the input perturbation approach, but has weaknesses in forms of NULL values, which could also reveal useful information to an adversary [2].

When a user requires deterministically correct answers, the QR method is the right answer. We already mentioned query restriction is Section 3. The idea of QR restriction is that when a query returns too much information it should not be revealed to the user. Deciding which query to allow and which to block is done using criteria techniques like Query Set Size Control (QSSC). The QSSC defined a threshold and blocks queries which affect a set of records which are smaller than the threshold. An example of such was presented in Section 3 when we proposed to block the answer of query 3, since it would return the value 1 and thus enable the user to infer sensitive data. A clear depiction of the QSSC process can be seen in Fig. 1. It demonstrates how QSS is partitioned from a table of data. For example if a user issues a query expressions [SEX = F AND DEPT = A AND B-YEAR = 1931] the partitioning tree corresponds to a unique path from the root to a leaf node, where only one leaf with such information hangs. In this case if the threshold is two, the query is denied.

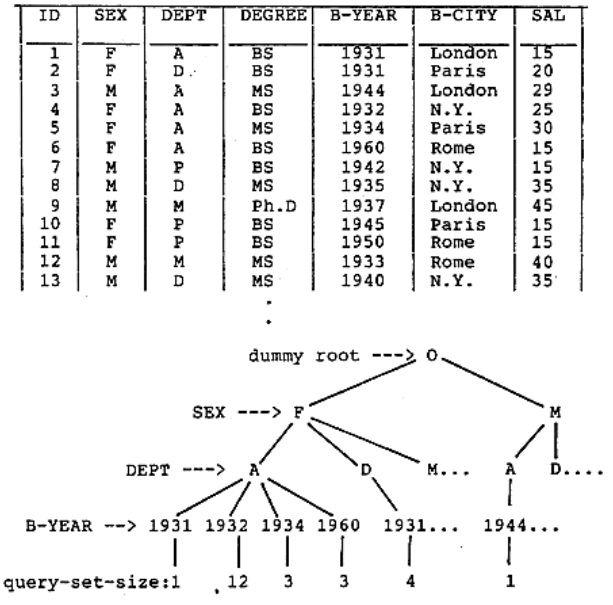

Figure 1. QSSC example - A partitioning tree of attributes SEX, DEPARTMENT, and B-YEAR [24].

Other methods of QR are Query Set Overlap Control (QSOC), Auditing, Partitioning, Cell Suppression, Implied Queries Control, Order Control, and Relative Table Size Control [2, 24]. QSOC suppresses all query responses which have more than a predetermined threshold of records in common with each prior response. Related to QSOC is the Partitioning method which in contrary to QSOC denies any query response with a slightest chance of an overlap. The partitioning is used to cluster the database into exclusive groups and a user can only query each group separately and as a whole and not a subset of the group [32]. A variation of partitioning is also micro aggregation which replaces clusters of sensitive data with their averages. The Auditing method is a specific method which works on the server side. The idea is that the server keeps a log of all queries issued and whenever a new one arrives, the server checks the stored queries and the new one combined to determine if an inference channel exists [2, 32]. This approach can brag with higher security since it takes all previous queries in consideration while checking for inference vulnerabilities. This is highly important since an adversary could use already gathered query responses in combination with new ones and construct an inference attack on his side without anyone noticing it. The auditing approach is also user friendly and ensures data truthfulness, since it provides users with normal query results as long as no inference vulnerability is sensed. The main disadvantage of the auditing approach method is it impracticality, since it requires a lot of computational power [2].

At last, when a user requires deterministically correct but not exact answers, Camouflage suffice as a method to protect sensitive information. The confidentiality via camouflage allows unlimited answers to any query type [22, 33]. The idea is to camouflage the confidential vector by making it part of a compact set of vectors.

4.3. Proactive inference control techniques

Inference control techniques can be divided according to the time of the inference detection-control process, i.e. during processing time (i.e. data level or reactive techniques) and during database design time (i.e. schema level or proactive techniques) [20]. The reactive technique or inference detection during processing time was already described above and acts in the way that if an inference channel is detected the query is either refused or the result is modified in order to avoid privacy violations (e.g. query restriction, perturbations, etc.). This approach can thus be applied to existing databases and uses data instances for its analysis. Although the reactive provides greater data availability it is thus more computationally costly because of all the process needed for enforcing inference control during processing time.

In contrary, the proactive approach allows database owners to deal with inference vulnerabilities through the use of special inference-oriented design

guidelines [20]. These techniques used to avoid inference vulnerabilities are applied during design time and do not need to access real data. The proactive inference control techniques were firstly studied in the late 1980s by Hinke [20] for relational databases and by Su and Ozsoyoglu [34] for multivalued dependencies. It was then that second path inference detection was developed, which uses only database schema analysis, without some deep knowledge, to detect possible inference channels. The concept is called second path because the analysis tries to reveal if any inferring path is possible by using various types of JOINS on database tables. Su and Ozsoyoglu presented a so-called Merlin schema analysis system which was used by the AERIE project as a mechanization of inference detection using second-path analysis based on functional dependencies [20].

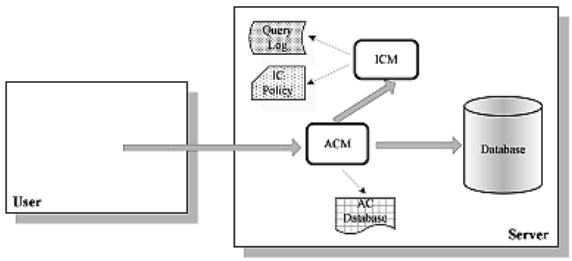

Figure 2. Traditional inference control architecture [2].

The AERIE project model was already described in Section 3. The problem with the proactive approach is the inflexibility and over classification of data [1]. The inflexibility comes into foreground when a database is already designed and deployed. Even if proactive inference detection at design time was conducted, it can happen that something was missed or neglected, or updates to the database schema take constantly place, etc. Such problems can be avoided by redesigning database schemas and by properly classifying attributes, but it is still unpractical und inflexible since changes usually also need to be made in applications and sometimes data gets duplicated and can lead to anomalies [19]. Furthermore, increasing the classification level of data can result in over-classification and thus reduce the availability of the same data.

Yip et al. showed that inference detection at schema level design level is inadequate [19, 35], meaning that although proactive inference control-detection approach is less computationally costly and user friendly than the reactive approach, the cost benefit ratio is still too high to rely only on it.

4.4. Inference Control Architecture

A depiction of a traditional inference control architecture can be seen in Fig. 2 and was well described in [2]. The architecture of inference control usually consists of multiple policy modules which regulate the security of the database. Typically the inference control architecture resides on the server where the database is located, although other possibilities were also presented [2].

In Fig. 2 we can see that a typical architecture consists of three basic blocks, i.e. Database, ACM and Inference Control Module (ICM). Regardless of the inference control architecture type an Access Control Module (ACM) is always present, as the first line of defense. This can be a DAC or a MAC access control module, and their job is to regulate the user access privileges and permissions before any of the inference control begins. Any user request to the server is always firstly directed to the ACM, where the user is crosschecked with the AC Database in order to verify his/her legitimacy and permission for a specific request. The Access Control Database stores for each user his credentials and permissions which are regulated by a server administrator. After user's request (i.e. query) was verified by the ACM it is then forwarded to the ICM for further inspection of any inference channel treat. In this example the ICM consist of two blocks, i.e. an IC policy and a Query Log. Each query which is directed to the ICM for inspection is firstly saved to the Query Log and then checked for treats using the Inference Control Policy (ICP). Depending on the type of the ICP, different strategies (e.g. QSSC, QSOC, etc.) of inference control are used in combination with all previously queries saved in the Query Log. If the ICP does not find any inference treat, the user's query is than marked as safe and revealed to the user.

5. Emerging Inference problem

In the last decade several new techniques attacking inference problem were proposed, e.g. using cardinality associations, incorporating imprecise and fuzzy database relations, employing functional dependencies, monitoring user activities, etc., but there is still no finite solution which covers the inference problem as a hole. Su and Ozsoyoglu showed that completely eliminating any precise inference is a NP-complete problem, marking it as a highly complex [36]. In addition, with the advancement of technology, including those of database structures, the rising numbers of collected data with the growing demand for data availability the inference problem shifts to the center of database security. New technology comes with new possibilities but also with undetected loopholes for possible exploitation, e.g. semi-structured databases, XML data structures, outsourcing, ontology, semantics, etc.

5.1. Encrypted Databases

A simple example of such precise problem involving inference is the outsourcing of databases and the security of those. Since many companies handle their own databases, the size of those increases during time and companies choose to outsource those to external providers, because of cost and security benefits [5]. The problem with such approaches is the question, how to ensure data confidentiality and integrity while still being able to use the database effectively. A solution for such a scenario is encryption of the database before outsourcing it to the external company. Unfortunately, encrypting the database, means the database cannot be used effectively, and for such reasons additional index information is added in order to maintain the availability and effectiveness of the outsourced data. The indexing process has to be done carefully, because indexing should be related to real data good enough to enable effectiveness while executing queries, but not open doors for possible inference and linking attacks. The right balance between index efficiency and protection is thus very important but also hard to achieve [5]. As with all inference problems, we have to assume that an adversary is highly motivated and has deep domain and prior knowledge. Damiani et al. describe two scenarios which should be considered about a possible adversary's prior knowledge in relation to an encrypted database [5]. The first scenario is when an adversary is aware of the exact or approximate distribution of values in the original-unencrypted database in addition to knowing the encrypted database, meaning that even if the adversary does not know the right index-value correspondence; he/she can determine it by comparing their occurrences (e.g. values with the same number of occurrences are indistinguishable). The second scenario is when the adversary has both versions of databases, i.e. the encrypted and the original, hence is able to use already mentioned inference attacks to reveal private information.

5.2. XML Databases and Semantic Web

Extensible Markup Language (XML) is a markup language that defines document encodings in a format readable by humans and machines, and a XML database is a data structure which stores data in the XML format. In the last few years the adoption of XML in professional and business environment has highly increased [7]. Like all database structures, XML database also have to deal with security, hence information stored could be private, sensitive or personal and thus need to be protected from revealing. Securing XML databases can thus be an important issue. A lot of scientific work emerged, which deals with access control for XML structures as a way of securing it, but none of it has addressed the problem of inference problem and the indirect access in XML structures [7]. The threats by inference in XML database are similar to other types of databases, but inference control methods for the traditional databases, which we described in the previous sections, cannot be applied on XML due to the flexibility of the formats and large-scale data availability. The structure of traditional (i.e. relational) databases is simple and fixed, hence structural information cannot be simply used for inferring sensitive information, whereby the structure of the XML databases is flexible and complex and labeled as ordered tree, making it easier for an adversary to use such structural information for inference [6].

Since data stored in today's databases has grown to vast dimensions it is often distributed or outsourced between multiple companies or organizations. Since XML databases are becoming more popular, some of this data is stored using such databases but with different XML formats or different security classifications. The problem with such scenarios emerge with the development of the Semantic Web, which raises new challenges in the security of XML, whereby ontology-based inference attacks emerge [7]. Adversaries can use the powerful reasoning abilities in order to disclose or infer sensitive information. An example of such is depicted in Fig. 3 where two XML documents have the same data but under different formats and security classification and as such could be used to induce ontology-based inference attacks.

XML Document 1 in Database A	XML Document 2 in Database B
<?xml version="1.0"?>	<?xml version="1.0"?>
<tool>……..*S*	<academic>……*P*
<document>……*S*	<paper>……..*P*
<title>CA109</title>	<title >CA109</title >
<author>John Smith</author>	<writer>John Smith</writer>
<project>P987HY5</project>	</paper >
</document>	</academic>
</tool>	

Figure 3.Example of two XML documents with the same data but under different format und security classification [7].

Some solutions for inference problem in XML databases were proposed, but mainly for XML documents with the same format [7, 37]. To the best of our knowledge, three solutions were presents to address the ontology-based inference attacks problem in XML databases, i.e. the Oxsegin by Stoica and Farkas, Oxicde by Zhang and Semsie by Yang et al. [7, 37, 38]. Both Oxsegin and Oxicde try to detect replicated data under conflicting classifications, using multiple XML documents and the relationship

in the ontology. After ontology equivalence between multiple XML documents has been done successfully, the indication of possible inference channels starts. The problem with both models is their inability to establish the correct correspondence between two elements under different syntax automatically, thus transferring the process to its users, making it highly time consuming. Semsie in contrary uses a multi-heuristic schema matching approach to deal with such problem, thus detecting possible inference channels automatically, without basing the detection on any manually pre-defined knowledge basis.

6. Conclusion

When dealing with database security, our thoughts are usually focused on the access control and physical server security, i.e. direct access, and seldom on the possible indirect access, i.e. on inference channel vulnerabilities. In this paper we presented the complexity and dimension of the inference problem in different database structures. We showed that implementing only access control in order to secure data has proven inadequate; although in practice security administrators often use only such static security techniques. We showed that the problem of completely removing all possible inference is NP-complete and that a complete inference detection model for all database structures does not exist. Although completely eliminating inference is impossible, we presented some promising inference control techniques which a security administrator should have in mind.

Furthermore, through the paper, we tried to present a profile of a possible adversary who is highly motivated and leverages the inference vulnerabilities by having high domain-knowledge and skills, and uses broad spectrum of available resources. It is very important for every database administrator, to have such an adversary profile in mind, while engaging with the security of a database.

In recent years new technologies like XML, Semantic Web and ontology, came to surface and present new dimensions in the inference problem. As we have shown in this paper some work in this field has already been done, but is still in the cradle. Further research possibilities for tackling the inference problem emerge could be addressed, e.g. inference channels after database updates, collusion group inference attacks, Ontology-based inference attacks, etc.

References

[1] V. Goyal, S.K. Gupta, M. Singh, A. Gupta. Auditing inference based disclosures in dynamic databases, Secure Data Management, Proceedings, 5159 (2008) 67-81.

[2] Y.J. Yang, Y.J. Li, R.H. Deng. New paradigm of inference control with trusted computing, Data and Applications Security Xxi, Proceedings, 4602 (2007) 243-258.

[3] C. Farkas, A. Brodsky, S. Jajodia. Unauthorized inferences in semistructured databases, Information Sciences, 176 (2006) 3269-3299.

[4] D.G. Marks. Inference in MLS database systems, Ieee Transactions on Knowledge and Data Engineering, 8 (1996) 46-55.

[5] E. Damiani, S.D. di Vimercati, S. Foresti, P. Samarati, M. Viviani.Measuring inference exposure in outsourced encrypted databases, Quality of Protection: Security Measurements and Metrics, (2006) 185-195.

[6] K. Hashimota, K. Sakano, F. Takasuka, Y. Ishihara, T. Fujiwara. Verification of the Security against Inference Attacks on XML Databases, Ieice Transactions on Information and Systems, E92d (2009) 1022-1032.

[7] L. Yang, J.M. Xiao, L. Jing. Protecting XML databases against ontology-based inference attack, Cis: 2007 International Conference on Computational Intelligence and Security, Proceedings, (2007) 838-842.

[8] R. Heatherly, M. Kantarcioglu, B. Thuraisingham. Preventing Private Information Inference Attacks on Social Networks, Knowledge and Data Engineering, IEEE Transactions on, PP (2012) 1-1.

[9] Y. Ishihara, S. Ako, T. Fujiwara. Security against inference attacks on negative information in object-oriented databases, Ieice Transactions on Information and Systems, E88d (2005) 2767-2776.

[10] B. Thuraisingham. Security and privacy for multimedia database management systems, Multimedia Tools and Applications, 33 (2007) 13-29.

[11] R. Los. McDonalds Database Compromise - 3rd Party Lessons, in, hp.com, 2010.

[12] R.W. Neal. Chinese Hackers Infiltrate US Army Database, Compromise Safety Of Thousands Of Dams, in: International Business Times, 2013.

[13] M.J. Schwartz. Sony Reports 24.5 Million More Accounts Hacked, in: InformationWeek Security, 2011.

[14] G. Flood. Sony Slapped With $390,000 U.K. Data Breach Fine, in: InformationWeek Security, 2013.

[15] G. Greenleaf. Global Data Privacy Laws: 89 Countries, and Accelerating, in: Queen Mary School of Law Legal Studies Research Paper No. 98/2012, 2012.

[16] T. Popeea, A. Constantinescu, L. Gheorghe, N. Tapus. Inference Detection and Database Security for a Business Environment, in: Intelligent Networking and Collaborative Systems (INCoS), 2012 4th International Conference on, 2012, pp. 612-617.

[17] T.S. Toland, C. Farkas, C.M. Eastman. The inference problem: Maintaining maximal availability in the presence of database updates, Computers & Security, 29 (2010) 88-103.

[18] A. Baraani-Dastjerdi, J. Pieprzyk, B.-d.J. Pieprzyk, R. Safavi-Naini. Security In Databases: A Survey Study, in, 1996.

[19] R.W. Yip, K.N. Levitt. Data level inference detection in database systems, 11th Ieee Computer Security Foundations Workshop - Proceedings, (1998) 179-189.

[20] T.H. Hinke, H.S. Delugach, R.P. Wolf. Protecting databases from inference attacks, Computers & Security, 16 (1997) 687-708.
[21] M. Hylkema. A Survey of Database Inference Attack Prevention Methods, in, Educational Technology Research, 2009.
[22] J. Domingo-Ferrer. Inference Control in Statistical Databases, in: L. Liu, M.T. Özsu (Eds.) Encyclopedia of Database Systems: Springer Science+Business Media LLC., 2009, pp. 1472-1476.
[23] G. Ozsoyoglu. On Inference Control in Semantic Data Models for Statistical Databases, J ComputSystSci, 40 (1990) 405-443.
[24] Y.H. Chin, P. Weng-Ling. An Evaluation of Two New Inference Control Methods, Software Engineering, IEEE Transactions on, SE-13 (1987) 1329-1339.
[25] S. Mandujano. Inference Attacks to Statistical Databases:Data Suppression, Concealing Controls and Other Security Trends, Aleph Zero online magazine, 5 (2000).
[26] J. Biskup, D.W. Embley, J.H. Lochner. Reducing inference control to access control for normalized database schemas, Information Processing Letters, 106 (2008) 8-12.
[27] T. Mulders, A. Storjohann. Diophantine linear system solving, in: Proceedings of the 1999 international symposium on Symbolic and algebraic computation: ACM, 1999, pp. 181-188.
[28] P. Samarati, L. Sweeney. Generalizing data to provide anonymity when disclosing information (abstract), in: Proceedings of the seventeenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems, Seattle, Washington, USA: ACM, 1998, pp. 188.
[29] V. Ciriani, S.D.C.d. Vimercati, S. Foresti, P. Samarati. k -Anonymous Data Mining: A Survey, Privacy-Preserving Data Mining, 34 (2008) 105-136.
[30] X. Sun, H. Wang, J. Li. Microdata protection through approximate microaggregation, in: Proceedings of the Thirty-Second Australasian Conference on Computer Science - Volume 91, Wellington, New Zealand: Australian Computer Society, Inc., 2009, pp. 161-168.
[31] J.F. Gonzalez, L.H. Cox. Software for tabular data protection, Statistics in Medicine, 24 (2005) 659-669.
[32] N.R. Adam, J.C. Worthmann. Security-control methods for statistical databases: a comparative study, ACM ComputSurv, 21 (1989) 515-556.
[33] R. Gopal, R. Garfinkel, P. Goes. Confidentiality via Camouflage: The CVC Approach to Disclosure Limitation When Answering Queries to Databases, Oper Res, 50 (2002) 501-516.
[34] T.-A. Su, G. Ozsoyoglu. DATA DEPENDENCIES AND INFERENCE CONTROL IN MULTILEVEL RELATIONAL DATABASE SYSTEMS, in, 1987, pp. 202-211.
[35] Y. Chen, W. Chu. Database Security Protection Via Inference Detection, in: S. Mehrotra, D. Zeng, H. Chen, B. Thuraisingham, F.-Y. Wang (Eds.) Intelligence and Security Informatics: Springer Berlin Heidelberg, 2006, pp. 452-458.
[36] J. Hale, S. Shenoi. Analyzing FD inference in relational databases, Data & Knowledge Engineering, 18 (1996) 167-183.
[37] A. Stoica, C. Farkas. Ontology guided XML security engine, Journal of Intelligent Information Systems, 23 (2004) 209-223.
[38] Z. Zhang. Research on Theory and Techniques for Preventing Inference Disclosure in Sensitive Databases, in: Institute of Communication Engineering, China: PLA Univ. of Sci. and Tech., 2006.

Corresponding author: M. Turkanović
Institution: University of Maribor, Faculty of Electrical Engineering and Computer Science
E-mail: muhamed.turkanovic@gmail.com