# Database Deductive Access Control

Xiaoqi Ma
School of Science and Technology
Nottingham Trent University
Nottingham, England, United Kingdom
Email: xiaoqi.ma@ntu.ac.uk

***Abstract*—Real data used in research projects give rise to conflicts of protecting privacy and providing more function: functionality requires trying to make use of maximum amount of data, while privacy requirements try to protect such information from being leaked. In this paper, a new model is proposed to capture relationships between database attributes and to record users' knowledge about database so as to prevent them from getting sensitive information by combining results returned from multiple queries, aiming to protect database privacy.**

***Index Terms*—database security, access control, deductive access**

## I. INTRODUCTION

With the rapid development of information technology, more and more research projects utilize real data. As the combination of private data with research provides more opportunities, it also poses a new dilemma: the balance of functionality and the concerns of privacy. Functionality requires trying to make use of maximum amount of data to support projects, while privacy requirements try to protect such information from being leaked and therefore protect privacy. In the United Kingdom, such kind of protection is derived from and emphasized in the Human Rights Act of 1998 [1] and the Data Protection Act of 1998 [2].

As an example, in the United Kingdom, the responsibility of people's health care is mainly undertaken by the National Health Service (NHS), which is composed of a large number of hospital trusts and other related organizations. In the NHS structure, each hospital trust is an independent legal entity and is responsible for the data it holds with respect to the principals of the Caldicott Report [3], which include (but are by no means limited to) the following basic rules:

(1) *Justify the purpose(s)*. Every proposed use or transfer of patient-identifiable information within or from an organization should be clearly defined and scrutinized, with continuing uses regularly reviewed, by an appropriate guardian.

(2) *Do not use patient-identifiable information unless it is absolutely necessary*. Patient-identifiable information items should not be included unless it is essential for the specific purpose(s) of that flow. The need for patients to be identified should be considered at each stage of satisfying the purpose(s).

(3) *Use the minimum necessary patient-identifiable information*. Where use of patient-identifiable information is considered to be essential, the inclusion of each individual item of information should be considered and justified so that the minimum amount of identifiable information is transferred or accessible as is necessary for a given function to be carried out.

(4) *Access to patient-identifiable information should be on a strict need-to-know basis*. Only those individuals who need access to patient-identifiable information should have access to it, and they should only have access to the information items that they need to see. This may mean introducing access controls or splitting information flows where one information flow is used for several purposes.

(5) *Everyone with access to patient-identifiable information should be aware of their responsibilities*. Action should be taken to ensure that those persons who are handling patient-identifiable information – both clinical and non-clinical staff – are made fully aware of their responsibilities and obligations to respect patient confidentiality.

From the above constraints the contradiction between functionality and privacy can be clearly seen. To provide more functions, more data tend to be exposed to users, and then the risk of compromising confidential information inevitably rises. Meanwhile, to better protect patients' personal and medical information, some functions have to be reduced from systems [4,5,6].

A lot of research work has been conducted on this issue and some valuable solutions have been proposed. One of the earliest solution is Stonebraker's query modification method [7,8,9]. Stonebraker's work used old fashioned QUEL query language and only had limited support to joined queries. Power *et al*. improved that by allowing joining, grouping and aggregation, and by implementing the method in SQL [10]. A formal framework for reasoning about inferences of query modification was also given by Power *et al*. in [11].


doi:10.4304/jcp.6.5.1024-1031

Another category of solutions involves *trackers*. The concept of tracker was thoroughly described in Denning *et al*.'s work [12,13,14]. Simpson's work combined both query modification and tracker [15].

Some others' work is also closely related to this issue although not directly intended to solve the specific problem. Dobkin *et al*. were among the earliest researchers working on secure database against malicious users' inference [16, 17]. A precise model about how attackers combined queries to infer sensitive information from database was presented in [16], although no complete counter measure was proposed. Sicherman *et al*. proposed a model which answered queries without revealing secrets by refusing to answer some certain questions [18]. Their model was rather theoretical and general, and "lacked of finiteness conditions". The model was quite secure but sometimes overkilling since it might simply refuse the whole query when it detected anything insecure, and it was difficult to implement. Toland *et al*. Analyzed the problem [19,20,21] and a remarkable model called *dynamic disclosure monitor* was described by them in [19,20]. This model maintained a history database and formed an index on it to store users' knowledge and therefore to resist inference attacks. The model was "sound and complete", but it "unnecessarily examined the entire history database in computing inferences" [19].

Byun proposed a secure anonymization technique dealing with data anonymization for incremental database [22], which was based on its "static" version proposed by Sweeney [23]. Sweeney's solution includes a formal protection model named *k*-anonymity and a set of accompanying policies for deployment; a release provides k-anonymity protection if the information for each person contained in the release cannot be distinguished from at least *k*-1 individuals whose information also appears in the release [23]. Both of these two techniques involved some kind of information loss or data distortion, which is not suitable to some applications.

In this paper, a new semantic model is proposed to capture relationships between attributes across multiple database tables and to record users' knowledge about the database. This way, if a malicious user tries to make multiple related legal queries and to combine the returned information together to figure out secret information, his knowledge will be recorded each time he issues such a query. Each time a new query is committed, the system will try to figure out what information can be deduced from this user's previous knowledge and the results of current query. Whenever the current query's results together with the user's previous knowledge compromise privacy, the system will immediately detect that and then reject the query.

## II. BACKGROUND

A relational database storing sensitive information can be attacked in different ways. Most attacks can be resisted by using various technologies, as described in literature. One special kind of attack involves deduction and inference. Attackers do not complete an attack in a single query; instead, they make several "legal" queries and then combine the results of them to deduce sensitive information which should not be allowed to disclose to them.

### A. A Sample Database

Suppose we have a database storing students and exams information. The database basically comprises two tables, namely the STUDENT table and the EXAM table.

The STUDENT table describes students' personal information. The schema of this table is as follows:

*STUDENT(SID, Name, DoB, Subject, Gender, Country, Club)*

In this schema, *SID* is the student's unique identifier and acts as a primary key; *Name* is the student's name; *DoB* is their date of birth; *Subject* is their subject area; *Gender* has two possible value, *M* and *F*; *Country* is where they are from; *Club* is the club they belong to.

Here is a sample populated table of STUDENT:

(S01, Alice, 07/09/1986, Math, F, Greece, Bridge)

(S02, Bob, 23/02/1990, Physics, M, Germany, Chess)

(S03, Cox, 17/10/1991, Psychology, M, Spain, Rugby)

(T20, Diana, 01/07/1992, Chemistry, F, Norway, Tennis)

(T30, Eva, 19/11/1985, Biology, F, Spain, Cycling)

(T40, Fred, 08/09/1993, Chemistry, M, Greece, Football)

The EXAM table describes students' exam information. The schema of this table is as follows:

*EXAM(SID, Date, Course, Result)*

The names of fields are self-explanatory. Each student may have multiple records in this table. The primary key of this table is {*SID*, *Date*}. In this table we suppose each student only take one exam in a single day (this restriction can be easily loosened by replacing date with a more precise time or using an extra identifier).

Here is a sample EXAM table:

(S01, 01/06/2009, Calculus, 78)

(T30, 02/06/2009, Molecular Biology, 85)

(S01, 03/06/2009, Statistics, 63)

(S03, 04/06/2009, Experimental Psychology, 90)

(T20, 04/06/2009, Organic Chemistry, 48)

(T30, 07/06/2009, Ecology, 62)

(S02, 07/06/2009, Mechanics, 75)

(T40, 07/06/2009, Inorganic Chemistry, 99)

(T30, 08/06/2009, Cell Biology, 87)

Suppose students' identities and their exam results are sensitive and should not be disclosed to any unauthorized users. The main concern on this database is that attackers

might have some way to deduce the identity of any student and to deduce personal and exam information about the victim. Such attacks compromise the victim's privacy.

Some mechanisms have been employed to prevent potential attackers from retrieving sensitive information from the database by using plausibly legal queries. Typically, the fields involved in a query will be analyzed to see whether knowing them will lead to leakage of sensitive information. If some secret information can be known by the attacker after a query, the query should either be rejected or modified in order to protect the confidentiality of the data.

For example, the designers of the database may believe that it is possible for attackers to deduce the identity of some certain patient by knowing his subject and the club he belongs to, as a club might have a small number of members.

Therefore when an attacker tries to issue the following SQL statement:

```
SELECT Subject, Club FROM STUDENT
WHERE DoB >= '01/02/1990'
      AND DoB <= '28/02/1990';
```

This query statement should be intercepted and rejected without getting any data returned before passed on to be run by the database system so that the information of subject and club will not appear in the query result at the same time.

Some variants of such kind of mechanism will take the fields involved in WHERE clause into account, giving attackers less opportunities to compromise the confidentiality of database.

However, this mechanism (and some other similar mechanisms such as limiting the size of returned query results) can work well only on individual queries. It is difficult to prevent attackers from making several legal queries and then joining these query results outside the database (possibly with the help of "trackers" [12,13] or knowledge from external sources).

### B. A Possible Attack

Attackers can bypass these mechanisms by issuing multiple related legal queries and combining the returned information to figure out the identity of some certain student and then further deduce sensitive personal and exam information about him.

For example, the attacker may know that someone belongs to a certain club and know his subject. Then the attacker could make a pair of separate queries with the same condition.

First of all, he issues the following SQL query statement, trying to retrieve the subject and other information of the intended person (suppose that the users are allowed to query subject, date of birth and gender information at the same time; otherwise they can choose other fields):

```
SELECT Subject, DoB, Gender FROM STUDENT
WHERE DoB >= '01/02/1990'
      AND DoB <= '28/02/1990';
```

And then he tries to fetch the club information using the following query statement:

```
SELECT Club, DoB, Gender FROM STUDENT
WHERE DoB >= '01/02/1990'
      AND DoB <= '28/02/1990';
```

If the attacker does not know the date of birth of the victim, these two queries will not contain the information of the intended victim as he expected. The attacker will be aware of this situation since the expected information does not appear in the query result. The attacker can then keep changing the period of date of birth in the query until the expected information appears in the results of both queries.

After that, the attacker should be able to join the results of the queries and get the identity of the victim, with which he can further query sensitive exam data of the victim. To resist such kind of attacks, the relationship between fields and the knowledge of attackers should be studied in depth.

## III. Semantic Metamodel

### A. The Semantic Database Model

The semantic model of the database describes database tables and the relationship of fields in each table. This model helps us get better understanding of the semantic structure of databases.

A database can consist of one or more tables. The set of all table names in the database is defined as TABLE. Note that TABLE only consists of tables containing user data. In the later parts of this paper, a number of administrative databases will be introduced into our framework. These tables should not be included in TABLE.

All the fields of tables in TABLE constitute the set FIELD. Usually there are more than one table in a database; different tables may have fields with same names. As an instance, both STUDENT and EXAM tables have a field called SID. To differentiate fields with same names in different tables, we identify a field by using the name of the table it belongs to as the prefix. For example, the notations STUDENT.SID and EXAM.SID can be used to differentiate two SID fields in two tables. When there is no ambiguity such as all other fields in these two tables, the prefix can be omitted for the sake of brevity.

Tables of database can be viewed as *sets of fields*, each of which comprises all fields of a table and is a subset of FIELD. We define a function Fields mapping from names of tables to their sets of fields:

$$\text{Fields}: \text{TABLE} \rightarrow P(\text{FIELD})$$

where $P$(FIELD) is the power set of FIELD.

A *critical* set of a table includes all fields within this table which should not be disclosed under any circumstances. If the values of any such a field are disclosed, students' identifies could be compromised. We define a function Critical mapping from names of tables to the corresponding critical sets as follows:

Critical: TABLE → $P$(FIELD)

A *conflict* set includes the minimal sets of anonymously conflicting fields. By "anonymously conflicting" we mean if the attacker knows the values of such a set of fields, it is possible for him to infer the identities of corresponding students. By "minimal" we mean any subset of an element of conflict set is not anonymously conflicting. In the definition of conflict set, we exclude all elements of the critical set from any element of conflict set, since disclosure of any single element of the critical set implies compromising students' identities. A function Conflict mapping from names of tables to the corresponding conflict sets is defined as follows:

Conflict: TABLE → $P(P$(FIELD))

A conflict set of a table can be empty.

An *inference set* indicates that a user can infer some information (in one field) from other information (in other fields). This is similar to the concept of functional dependence. The difference between inference set and functional dependence is that the latter is a "total dependence", that is to say that all values in one or more columns functionally decide the values in other fields, while our inference set means that such dependence could be either partial or total. In the concept of partial inference, some values in one or more fields decide values in another field, while other values in the same field or fields do not. We define a function Inference as follows:

Inference: TABLE → $P(P$(FIELD) $\Longrightarrow$ FIELD)

where $\Longrightarrow$ could be either the symbol of total inference (denoted as $\Longrightarrow_T$) or partial inference (denoted as $\Longrightarrow_P$).

In the case of partial inference, a separate table is needed to describe what values of the field(s) can be used to infer information of other fields for each combination of table and user whose inference set contains partial inferences.

### *B. Knowledge Records of Database*

Although a number of mechanisms have been employed to protect privacy of databases, it is still possible for attackers to use some techniques such as trackers to infer confidential information by issuing a series of legal queries and then joining the query results together outside the database. To prevent such kind of attacks, we can try to record what has been known by the user, and when he issues a new query, we can check the records of this certain user and the results of the new query to figure out whether he can infer confidential information from them. If it is possible for the user to compromise privacy of the database, we should reject the new query.

The knowledge information of users is called *knowledge records*. The knowledge records of each table can be stored in *knowledge tables*. A knowledge table contains all "visible" fields as well as the primary key of the table.

### *C. Making Decisions Based on Knowledge Records*

The semantic model and knowledge records can be combined to resist attacks described above. To achieve this, we need to follow the following steps:

1. Analyze the fields of the query issued by the user to see whether the query itself is acceptable or not based on the critical set, conflict set and inference set. If not, reject the query; otherwise go to next step.
2. Conduct the intended query, based on which produce the query result as well as the list of the primary key values of the query results.
3. Based on the knowledge table produce the knowledge view $V_O$ (old knowledge).
4. Combine the results produced in step 2 and $V_O$ to see whether the new query will compromise the identities of students. If so, the query should be rejected; otherwise, return the query results to the user and update the knowledge records accordingly.

In the first step, as well as the fields appearing in the SELECT clause of the SQL statement, the fields appearing in the WHERE clause should also be counted into the set of "query fields". If any subset of the query fields set totally decides other fields (using the total inferences in the inference set to decide), these decided fields should also be included into the query fields set. This process should be repeated until the query fields set does not expand any more. If any element of the critical set is also an element of the expanded query fields set, or any element of the conflict set is subset of the expanded query fields set, this query should be identified as unacceptable.

In the last step, both total and partial inference should be considered. There are five different cases in the combination:

1. A row appears in $V_O$ but not correspondingly appears in the new query. This means that the user will not be able to get extra information on this student. The privacy of this row will not be compromised, and we do not have to do anything on this.
2. A row is returned within the query result, but does not appear in $V_O$. This is also safe, but we need to add the knowledge information into the knowledge table.
3. A row appears in both $V_O$ and the query result, and the combination of them will not compromise the privacy of the corresponding student. In other words, the user cannot get sensitive information on this row after issuing this query. In this case, the knowledge table should be updated.

4. A row appears in both $V_O$ and the query result, and the combination of them may or may not compromise the privacy of the corresponding student, depending on the values of the known fields. This case concerns partial inferences. The table recording partial inferences should be looked up to decide whether this query is safe or not.
5. A row appears in both $V_O$ and the query result, and the combination of them will compromise the privacy of the corresponding student. The query should be rejected, and the knowledge table should be left unchanged.

After these five steps, the system finishes its job for the current query and is ready for the next one. The returned query results (if any) will not compromise the confidentiality of the database.

## IV. Protecting the Sample Database: An Example

### A. The Semantic Model

As stated above, the semantic model of the database describes database tables and the relationship of fields in each table.

The TABLE set of the above sample database is

TABLE = {STUDENT, EXAM}

The FIELD set of the database is

FIELD = {STUDENT.SID, Name, DoB, Subject, Gender, Country, Club, EXAM.SID, Date, Course, Result}

Accordingly, the definition of the function Fields can be:

Fields(STUDENT) = {STUDENT.SID, Name, DoB, Subject, Gender, Country, Club}
Fields(EXAM) = {EXAM.SID, Date, Course, Result}

Suppose in STUDENT table the field Name should never be disclosed. So we have

Critical(STUDENT) = {Name}

The critical set of EXAM is empty.
A possible conflict set could be

Conflict(STUDENT) = {{Subject, Club}}

A conflict set of a table can be empty set, as the case of EXAM table.

As an instance of partial inference, some certain subjects only have students from a certain country, probably for a certain gender. For example, suppose all male students in politics are from Canada, all female students in math are from Greece, and all students in economics are from China. For all other subject and gender information, we cannot infer any information of country. For example it is impossible to infer a student's country of origin by just knowing that he is male student in chemistry. So we have

Inference(STUDENT)=
{{Subject,Gender}$\Rightarrow_{P}$Country}

In this example, a table is attached to this partial inference describing what kind of information of country of origin can be derived from subject and gender information.

### B. The Knowledge Records

A sample knowledge table of STUDENT looks like this:

(S01, NULL, Math, NULL, NULL, NULL)
(S02, NULL, NULL, M, NULL, Chess)
(S03, 17/10/1991, NULL, NULL, Spain, NULL)
(T20, NULL, NULL, F, NULL, Tennis)
(T30, 19/11/1985, NULL, NULL, NULL, Cycling)
(T40, 08/09/1993, NULL, M, NULL, NULL)

Note that the field Name does not appear the table as Name is in the critical set and should not be disclosed anyway.

The reason of storing the real values of known information in knowledge tables is that partial inferences are based on values, unlike functional dependence. However, for those fields which are not involved in partial inferences, we do not have to store the values for them. For example, student T30's club information has been known, therefore according to the conflict set, the information of country should not be disclosed, whatever the values these fields have. In such a case, recording the real values of Club is unnecessary. Also, since the fields DoB and Club are not involved in the value-based inference, the real values of them could be left off the knowledge table and we only need to record whether or not they have been known. Therefore the knowledge table can be simplified as follows:

(S01, No, Math, NULL, NULL, No)
(S02, No, NULL, M, NULL, Yes)
(S03, Yes, NULL, NULL, Spain, No)
(T20, No, NULL, F, NULL, Yes)
(T30, Yes, NULL, NULL, NULL, yes)
(T40, Yes, NULL, M, NULL, No)

### C. Making Decisions

With the above semantic metamodel and the knowledge table, we now can protect our sample database from malicious attacks.

Suppose the user's current knowledge table is as follows:

(S01, No, Math, NULL, NULL, No)
(S02, No, NULL, M, NULL, Yes)
(S03, Yes, NULL, NULL, Spain, No)
(T30, Yes, NULL, F, NULL, No)

This forms the user's $V_O$.

Suppose the user issues the following query on the STUDENT table:

**SELECT** Subject, DoB, Gender **FROM** STUDENT
**WHERE** Club = 'Rugby';

The set of query fields of this query is

{Subject, DoB, Gender, Club}

We call this set *F*.

Obviously the element of the conflict set of the STUDENT table {Subject, Club} is subset of *F*, therefore this query is not acceptable.

Suppose the user then issues another query

**SELECT** DoB, Gender, Club **FROM** STUDENT
**WHERE** DoB >= '01/01/1986'
**AND** DoB <= '31/12/1992';

This query is acceptable according to the above criteria. Then we conduct the intended query and produce the query results together with the primary key value of each row of the query results. In this example, four rows are returned, whose primary key values are S01, S02, S03 and T20:

(S01, 07/09/1986, F, Bridge)

(S02, 23/02/1990, M, Chess)

(S03, 17/10/1991, M, Rugby)

(T20, 01/07/1992, F, Tennis)

Now we can combine $V_O$ with the new knowledge produced by the query to see whether this query compromises the security of the database or not. To do so, we need to compare each row of the query result with the corresponding row in $V_O$. We can see five different cases in the comparisons:

1. The row of the student T30 appears in $V_O$ but not in the new query. The privacy of T30 will not be compromised, and we do not have to do anything.

2. The record of the T20 is returned within the query result, but it does not appear in $V_O$. It is also safe, but we need to add the following row into the knowledge table:

   (T20, Yes, NULL, F, NULL, Yes).

3. The row of S02 appears in both $V_O$ and the query result, and the combination of them will not compromise the privacy of this student. Therefore it is also safe, but we need to update the knowledge status of the field DoB in the knowledge table.

4. The row of S01 appears in both $V_O$ and the query result, so we need to investigate the values of the fields known by the user. Before the query the user knows that student S01 is studying math. By issuing the query, the user will know S01's club and gender, which is female. So the set of fields the user will know about S01 after the query would be

   {DoB, Subject, Gender, Club}

   According to the inference set on STUDENT table, the user might be able to infer S01's country by knowing her subject and gender. In this example, we can conclude that since the user knows S01 is a female student in math, he would be able to infer that S01's country is Greece. Therefore the user will know the following set of fields about S01:

   {DoB, Subject, Gender, Country, Club}

   Since an element of the conflict set on STUDENT table is a subset of the table of known fields, this query will compromise S01's identity and should be rejected, and all previous changes to the knowledge table should be rolled back. On the other hand, if S01 is studying politics and all other information remains same, the query will be safe, since by knowing S01 is a female student studying politics, the user cannot infer S01's country of origin. Therefore in this revised case the query is acceptable, and we need to update the row of S01 in the knowledge table as follows:

   (S01, Yes, Politics, F, No, Yes).

5. The row of S03 appears in both $V_O$ and the query result, and the combination of them discloses S03's identity. Before the query the user knows S03's DoB and Country, and by the query he will know S03's DoB, Gender and Club information. In combination, the user will know the following set of fields about the record of S03:

   {DoB, Subject, Gender, Club}

   which contains an element of the conflict set on STUDENT table and compromises S03's identity. So this query should be rejected, and all changes to the knowledge table should be rolled back.

This example clearly shows that the new metamodel successfully resists attacks combining multiple legal query results.

## V. Conclusion and Future Work

In this paper, a new semantic metamodel has been proposed, enabling databases to capture relationships between database attributes and to record user's knowledge about databases. As a result, it could help databases resist attacks aiming to get sensitive information by combining results returned from multiple queries.

In the metamodel, several sets have been defined to capture the semantics of database attributes. The sets include tables set, fields set, critical set, conflict set and inference set. Some functions have also been defined upon these sets. To record user's previous knowledge

about the database, it has been suggested to maintain user's knowledge records. A standard method has also been described in this paper to help to decide whether to accept or decline queries and to update user's knowledge records. In this way, when an attacker tries to issue a query whose results could compromise the privacy of the database possibly by combining the results with his previous knowledge, it will be detected immediately, and the database will reject the latest query to protect students' privacy.

The metalmodel considers the previous knowledge of the user, while most methods described in Section I do not, and that is the reason why attacks concerning multiple queries can be effectively resisted by using this new model. Unlike the $D^2$Mon model, this new model store the knowledge history in a very compact way, therefore previous knowledge searching will be much more efficient.

The new metamodel only considers a single database on a single server. The situation of federated databases will be the topic of next step of research.

## ACKNOWLEDGMENT

The author thanks Andrew Simpson of Oxford University Computing Laboratory (OUCL) for his valuable advice and feedback.

## REFERENCES

[1] Human Rights Act 1998. The Stationery Office Limited, London, 1998.

[2] Data Protection Act 1998. The Stationery Office Limited, London, 1998.

[3] The Caldicott Report. http://www.publications.doh.gov.uk/ipu/confiden, December, 1997.

[4] A. Simpson, D. Power, M. Slaymaker, E. Politou, "GIMI: Generic infrastructure for medical informatics", *Proceedings of the 18th IEEE Symposium on Computer-Based Medical Systems*, Dublin, Ireland, pp. 564-566, 2005.

[5] C. Tromans, M. Brady, D. Pawer, M. Slaymaker, D. Russell and A. Simpson, "The application of a service-oriented infrastructure to support medical research in mammography", *MICCAI-Grid Workshop* – Olabarriaga, Lingrand and Montagnat (Eds.), pp. 43-51, 6 September 2008, New York.

[6] A. Simpson, D. Power, D. Russell, M. Slaymaker, V. Bailey, C. Tromans, M. Brady and L. Tarassenko, "GIMI: The past, the present and the future", *Philosophy Transactions of The Royal Society (A)*, (2010)368, pp. 3891-3905, 2010.

[7] M. Stonebraker and E. Wong, "Access control in a relational data base management system by query modification", *ACM/CSC-ER Proceedings of the 1974 annual Conference*, pp. 180-186, 1974.

[8] M. Stonebraker, "A functional view of data independence", *International Conference on Management of Data, Proceedings of the 1974 ACM SIGFIDET workshop on Data description, access and control*, pp. 63-81, Ann Arbor, Michigan, US, 1974.

[9] M. Stonebraker, "Implementation of integrity constraints and views by query modification", *Proceedings of the 1975 ACM SIGMOD International Conference on Management of Data*, San Jose, California, USA, pp. 65-78, 1975.

[10] D. Power, M. Slaymaker, E. Politou, and A. Simpson, "Protecting sensitive patient data via query modification", *Proceedings of the 2005 ACM Symposium on Applied Computing*, Santa Fe, New Mexico, USA, pp. 224-230, 2005.

[11] D. Power, M. Slaymaker, and A. Simpson, "On deducibility and anonymization in medical databases", *Secure Data Management – SDM2005*, volume 3647 of Lecture Notes in Computer Science, pp. 170-184, September 2005.

[12] D. E. Denning, P. J. Denning, and M. D. Schwartz, "The tracker: a threat to statistical database security", *ACM Transactions on Database Systems*, volume 4, issue 1, pp. 76-96, 1979.

[13] D. E. Denning and J. Schlörer, "A fast procedure for finding a tracker in a statistical database", *ACM Transaction on Database Systems*, Vol. 5, No. 1, pp. 88-102, March 1980.

[14] D. E. Denning, "Secure statistical databases with random sample queries", *ACM Transactions on Database Systems*, Vol. 5, No. 3, pp. 291-315, September 1980.

[15] A. Simpson, D. Power, and M. Slaymaker, "On tracker attacks in health grids", *Proceedings of the 2006 ACM Symposium on Applied Computing*, Dijon, France, pp. 209-216, 2006.

[16] D. Dobkin, A. K. Jones, and R. J. Lipton, "Secure databases: protection against user influence", *ACM Transactions on Database Systems*, volume 4, issue 1, pp. 97-106, 1979.

[17] R. Demillo, D. Dobkin, R. Lipton, "Even data bases that lie can be compromised", IEEE Transactions Software Engineering. SE-4, 1, pp. 73-75, 1978.

[18] G. L. Sicherman, W. de Jonge, and R. P. van de Riet, "Answering queries without revealing secrets", *ACM Transactions on Database Systems*, volume 4, issue 1, pp. 41-59, 1983.

[19] T. S. Toland, C. Farkas, and C. M. Eastman, "Dynamic disclosure monitor ($D^2$Mon): an improved query processing solution", *Secure Data Management – SDM2005*, volume 3647 of Lecture Notes in Computer Science, pp. 124-142, September 2005.

[20] C. Farkas, T. S. Toland, and C. M. Eastman, "The inference problem and updates in relational databases", *Proceedings of the Fifteenth Annul Working Conference on Database and Application Security*, Niagara, Ontario, Canada, pp. 181-194, 2001.

[21] A. Brodsky, C. Farkas and S. Jajodia, "Secure Databases: Constraints, Inference Channels and Monitoring Disclosures," *IEEE Transactions on Knowledge and Data Engineering*, 12(6): 900-919, 2000.

[22] J.-W. Byun, Y. Sohn, E. Bertino, and N. Li, "Secure anonymization for incremental datasets", *Secure Data Management – SDM2006*, volume 4165 of Lecture Notes in Computer Science, pp48-63, 2006.

[23] L. Sweeney, "k-anonymity: a model for protecting privacy", *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, volume 10, issue 5, pp. 557-570, 2002.

**Xiaoqi Ma** graduated from Nanjing University of Science and Technology, China, with a first class BSc degree in Computer Science in 1997. He received his MSc degree in Computer Science from the Institute of Software, Chinese Academy of Sciences in 2003. His PhD degree, also in Computer Science, was obtained from the University of Reading, UK, in 2007.

He used to work in Beijing Red Flag Software Co., Ltd., as a research associate when he did his Msc research. He served as senior software engineer in Dwight Cavendish System (UK) for a short period in 2003. He worked in Oxford University Computing Laboratory as a Research Officer from October 2007 to February 2009. He has been a Lecturer/Senior Lecturer in Computer Science in School of Science and Technology, Nottingham Trent University, Nottingham, UK, since March 2009.