A Unified Audit Expression Model for Auditing SQL Queries

Vikram Goyal[1], S. K. Gupta[1], and Anand Gupta[2]

[1] Department of Computer Science and Engineering, IIT Delhi
Hauz Khas, New Delhi-16
(vkgoyal, skg)@cse.iitd.ernet.in

[2] Dept. of Comp. Sci. and Engg. N.S.I.T. Delhi, Sector-3, Dwarka,
New Delhi
anand@coe.nsit.ac.in

Abstract. A privacy auditing framework for Hippocratic databases accepts an administrator formulated audit expression and returns all suspicious user queries that satisfy the given constraints in that audit expression. Such an expression should be expressive, precise, unambiguous and flexible to describe various characteristics of a privacy violation such as target data (sensitive data subject to disclosure review), suspicion notion, authorized privacy policy parameters through which the violation is possible, and time duration of the privacy violation. Earlier proposed audit expression models for the auditing are not flexible and do not specify suspicion notion with in the audit expression for the auditing of past user accesses. We propose a unified model for an audit expression which can specify earlier proposed audit expressions along with different suspicion notions. The model includes (i) a suspicion notion model which unifies earlier proposed suspicion notions, and (ii) mechanisms to specify data versions.

1 Introduction

Privacy concerns have become very prominent in e-commerce, e-governance and a host of services delivered through the Internet. Governments have enacted regulatory laws balancing various needs to provide robust and acceptable privacy. Academic and commercial organizations have also carried out research to achieve the holy grail of complete privacy. However, despite considerable efforts [1–4], privacy intrusions [5–9] continue to rise and raise serious concerns. Providing robust privacy infrastructure remains an elusive and perhaps an Utopian goal.

Faced with the problem of privacy violations, the next step in providing confidence to the involved parties is to detect privacy violation accesses [10–12]. In case of privacy violation notification, it is required to determine the source of the privacy violation, i.e., to ask the questions like who? when? and how? etc. This process of privacy violation access determination is termed as *auditing*. Auditing may lead to: (a) putting some individuals under suspicion, (b) acting against those involved in that violation, if confirmed, and (c) locating and fixing the specification or implementation loopholes in the privacy or access control policies. These findings and their maintenance increase the trust of involved parties in the organization and the information system.

In a post event scenario, the auditing process generally starts from the available information related to privacy violation. Therefore, the need for some specification mechanism for this information in an expression is evident. The information which may be available are: (a) target data view which has caused a privacy violation, (b) notion of suspicion, (c) time interval of attack, and (d) privacy policy specific parameters.

Prior work in the area of SQL query auditing does not facilitate specifying different suspicion notions for the administrator but instead assumes a default suspicious notion. A suspicion notion defines the criterion by which suspicion of a batch of queries is determined. Earlier work includes Agrawal et al. [12] where a simple specification syntax and a notion of suspicion was introduced for single SQL queries in isolation, Motwani et al. [13] where the authors have used a similar syntax and have proposed new suspicion notions for a batch of queries, i.e., semantic suspiciousness, strong syntactic suspiciousness, and weak syntactic suspiciousness. Other work in auditing is Böttcher et al. [14] where the authors have used audit expression for XML databases and used the similar notion of suspicion as proposed by Agrawal et al. in [12]. Consider the following example of an audit expression:

```
AUDIT disease FROM Patients WHERE zipcode='118701'
```

The audit expression asks for auditing of disease information for all the patients living in area 118701. It assumes the default suspicion notion of *indispensable tuple* (formally explained later) for determining suspiciousness of a query, i.e., it marks a query suspicious if it has accessed disease information of at least one patient from the patients which get identified by the audit expression. On the other hand, there may be many other suspicion notions here, e.g., (i) access to disease and area information of at least one patient from the above identified patients, (ii) access to disease information of more than *N* patients from the above identified patients, and many more. Therefore, the earlier proposed specification syntax is simple but is not expressive enough to specify different suspicion notions. It makes the administrator's task of specifying audit expressions difficult as different privacy violations may require different suspicion notions. Hence, we propose a suspicion model which is capable of expressing different interesting suspicion notions.

Further, auditing generally occurs on past user accesses for a given time interval. In case of database applications, where the database state changes frequently due to various *update* and *insert* operations, two identical queries issued at different times might have accessed different information. As an audit expression specifies the target data, e.g., the disease information of the patient living in area *118701* in the given example, there may be many versions of the data for a given time interval. Existence of such different instances of database emphasizes the need to incorporate version specific information in the audit expression and hence is incorporated in our model. The administrator also specifies the privacy policy specific information in an audit expression to limit the search for intended user accesses [12, 15, 10], which is also supported in our model.

The motivation for proposing such an audit expression model arises from the need of representing different information in an audit expression by the auditor while auditing, as it is usually done in the context of information search over the Internet. A user usually gets some feedback from the result of first query and tries to embed that information obtained from the result in the next query to get the relevant results. Similarly, the proposed model helps to retrieve relevant suspicious queries as the auditor can express the required information in the proposed audit expression model which was not possible in the earlier cases. In this paper, we contribute the following:

1. Identify components for an audit expression that need to be specified for auditing.
2. A model for suspicion notion which can encompass the earlier defined notions of suspicion as well as other relevant suspicion notions.
3. Incorporates proposed components in an audit expression,
   - giving the flexibility of specifying suspicion notions,
   - specifying precisely target data view in presence of more than one data versions, and
   - specifying precisely limiting information parameters such as privacy policy parameters.

The rest of the paper is organized as follows. In Section 2, related work is presented. Section 3 presents the audit expression model. In Section 3.1, we describe target data view specification in presence of multiple data versions. We present the suspicion model in Section 3.2, limiting parameters for auditing in Section 3.3, and audit expression in Section 3.4. Finally in Section 4, we conclude with suggestions for future work.

2 Related Work

In this section, we describe earlier audit related work done by different authors.

2.1 Data Dependent Auditing of SQL Queries

In [12] Agrawal et al. explore the auditing problem of determining whether any single SQL query in the query log accessed a specific information specified by an audit expression. Their syntax for audit expressions (Figure 1) closely resembles SQL queries.

```
OTHERTHAN PURPOSE   purpose list
DURING              timestamp 1 to timestamp 2
AUDIT               attribute list FROM table names WHERE conditional expression
```

Fig. 1. Audit Expression Syntax as proposed by Agrawal et al. [12]

During normal operation, the text of every query processed by the database system is logged along with annotations such as the execution time, the user-id of a user who submits the query, and the purpose (defined in [4]) for which the query is made by the user. The system uses database triggers to capture and record all updates to base tables in backlog tables. The state of the database at any past point in time is obtained through these backlog tables.

For the audit, the auditor formulates an audit expression that declaratively specifies the data of interest. The audit expression is processed by the audit query generator. It first performs a static analysis over the logged queries to select a subset of logged queries that could potentially disclose the specified information for the given audit expression. The query generator then analyzes the selected queries by running them against the backlog database and yields the precise set of logged queries that accessed the designated data.

An audit expression essentially identifies the tuples of interest via predicates in the *WHERE* clause from the cross-product of tables in the *FROM* clause. Any query which has accessed all the attributes in the audit list and the *WHERE* clause of which gets satisfied by any of the identified tuples is tagged as a suspicious query. We illustrate this with examples from [12]. Consider the audit expression:

```
AUDIT disease FROM Patients WHERE zipcode='120016'
```

This expression tags all queries that returned disease information about any patient living in area *120016*. Now consider the SQL query:

```
SELECT zipcode FROM Patients WHERE disease='cancer'
```

This SQL query will be considered suspicious with respect to the above audit expression if any patient who has *cancer* lives in area *120016*. It would not, however, be suspicious with respect to the following expression if no patient having both *cancer* and *diabetes* exists.

```
AUDIT zipcode FROM Patients WHERE disease='diabetes'
```

This is due to the fact that this audit expression checks only whether the zipcode of any patient with diabetes was disclosed.

Looking at the version related issue, it may be noted that the above expression identifies tuples in the current database instance only, whereas the authors in [12] interpret it as all the versions of zip codes of each *diabetes* patient present in the backlog database table (we use *b-relationname* to specify the backlog table of table *relationname*, e.g., b-Patients for table Patients), whereas the authors in [13] interpret this as zipcode of each diabetes patient in the Patients table of the current database instance, i.e., Patients table. Both interpretations would give different results if there had been updates of zipcode and disease value of a patient. We handle these version related issues in our proposed audit expression model.

It may be noted that all the above examples assumes a suspicion notion of *indispensable tuple* [12] (explained later). Similarly, many other suspicion notions have been proposed in the past for auditing. To explain these notions, we use following formalizations. The SPJ (Select Project Join) queries and audit expressions are considered of the form $Q = \pi_{C_{OQ}}(\sigma_{P_Q}(\mathcal{T} \times \mathcal{R}))$ and $A = \pi_{C_A}(\sigma_{P_A}(\mathcal{T} \times \mathcal{S}))$ respectively. Here $\mathcal{T}$ is the cross-product of tables common to both the audit expression and the query. $\mathcal{R}$ and $\mathcal{S}$ are the cross-products of the other tables in the *FROM* clause. C_{OQ} is the set of columns projected out in query Q and C_A is the set of columns to be audited in A. P_Q and P_A are the predicates in Q and A respectively specified in *WHERE* clause. We use C_Q to denote all the column names appearing in C_{OQ} and P_Q.

Definition 1. *(Candidate Query) A query Q is a candidate query with respect to an audit expression A, if Q can not be marked syntactically non suspicious with respect to an audit expression A for a given suspicion notion. By syntactically we mean that query and audit expression are not executed over any database instance.*

Definition 2. *(Indispensable Tuple) A tuple $t \in \mathcal{T}$ is indispensable to a query Q if the presence or absence of tuple t creates a difference to the result of the query Q, i.e. $\sigma_{P_Q}(t \times \mathcal{R}) \neq \phi$.*

Definition 3. *(Notion of Semantic Suspicion defined in [12] for a single query) A candidate query Q is suspicious with respect to an audit expression A if both share an indispensable tuple. A candidate query is the query which accesses all the columns listed in the audit expression, i.e., $C_Q \supseteq C_A$.*

Definition 4. *(Notion of Semantic Suspicion defined in [13] for a batch of queries) A batch of queries $\mathcal{Q}$ is said to be semantically suspicious with respect to an audit expression A if there is some subset of queries $\mathcal{Q}' \subseteq \mathcal{Q}$ such that (1) there is a tuple $t \in \mathcal{T}$ for every query $q \in \mathcal{Q}'$ that is indispensable to both q and A, and (2) the queries in $\mathcal{Q}'$ together access all the columns of the audit list in A. Here $\mathcal{T}$ is the cross product of all tables common to A and the particular $q \in \mathcal{Q}'$ in question.*

2.2 Data Independent Auditing of SQL Queries

This type of auditing is done independent of a database instance, i.e., a database instance is not accessed. Due to being independent from a database instance this would be very fast as compared to database dependent auditing as accessing a database is a costly operation. But, unfortunately, it is computationally intractable to determine suspicion for many query types for a given audit expression and a notion of suspicion [16, 17, 13].

The authors in [16, 17] have considered the problem of “perfect privacy” which determines whether a database system discloses *any information* at all about a secret view through various views revealed by it. Here secret view corresponds to the audit expression and the views that were revealed to answered queries. Determining whether “any information” has been disclosed is determined through a notion of suspicion which uses the *critical tuple* concept.

Definition 5. *(Critical Tuple [17]) A tuple $t \in D$, where D is the database, is critical for a query q if there exists a possible database instance I of D for which $q(I-t) \neq q(I)$, i.e., t is critical for q if there exists some instance for which dropping t makes a difference to the result of q.*

Definition 6. *(Perfect Privacy Suspicion Notion [17]) An SQL query q is suspicious with respect to a secret view A, if and only if both q and A share a critical tuple.*

Definition 7. *(Notion of Weak Syntactic Suspicion given in [13]) A batch of SPJ queries $\mathcal{Q}$ is weakly syntactically suspicious with respect to an audit expression A, if there exists some subset of the queries $\mathcal{Q}' \subseteq \mathcal{Q}$ and some database instance I such that (1) for every query $q \in \mathcal{Q}'$, a tuple $t \in \mathcal{T}$ is indispensable to both q and A in the context of I and (2)* *the queries Q' together access at least one of the columns of the audit list in A. Here T is the cross product of all the tables in I common to both A and the query $q \in Q'$ under consideration.*

The difference between perfect privacy suspicion notion and weak syntactic suspicion notion is that the later notion requires accessing of at least one column of audit query by a user query to be tagged as suspicious.

2.3 Auditing Aggregate Queries

The problem of auditing aggregate queries has been extensively studied in the context of statistical databases [18]. Users of statistical databases can retrieve only aggregate results. In this paper, we consider only SPJ queries for relational databases and our work is orthogonal to the body of work done for statistical databases.

3 Audit Expression Model

It may be noted from the previous section that in the auditing research paradigm, several suspicion notions have been proposed by different authors with different objectives but no one has worked for the incorporation of the suspicion notion with in an audit expression. In this section, we present a suspicion model and show that our proposed suspicion model can specify all the above suspicion notions (defined in Section 2) in addition to other relevant suspicion notions. We incorporate the proposed suspicion model in the audit expression which increases the expressiveness of the proposed audit expression. Further, we introduce the notion of data versions specification in the audit expression which helps the auditor to specify precise and unambiguous audit expression.

The model consists of (i) a target data view, (ii) suspicion notion, and (iii) filtering parameters. The target data view describes the sensitive data which is under disclosure review. The suspicion notion identifies the portions of the target data view which, if accessed by a batch of queries make it suspicious. Filtering parameters are the constraints specifying context information such as the time interval of user accesses. Now we shall explain each of these constituents in the following subsections. The following relations (Tables 1, 2, 3) and audit expressions (Figures 2, 3) are used for describing our proposed model.

t-id	pid	name	age	sex	zipcode	address
t11	p1	Jane	25	M	177893	A1
t12	p2	Reku	35	M	145568	A2
t13	p13	Robert	29	M	188888	A3
t14	p28	Lucy	20	F	145568	A4

Table 1. P-Personal

t-id	pid	ward	doctor	disease	pres-drugs
t21	p1	W23	Hassan	thyroid	drug2
t22	p2	W12	Nicholas	diabetic	drug1
t23	p13	W14	Ramesh	Malaria	drug3
t24	p28	W14	King U	diabetic	drug1

Table 2. P-Health

tid	pid	employer	salary
t31	p1	E1	12000
t32	p2	E2	20000
t33	p13	E3	9000
t34	p28	E4	19000

Table 3. P-Employ

```
Audit name,age,address FROM P-Personal WHERE age < 30
```

Fig. 2. Audit Expression-1

```
Audit name, disease, address FROM P-Personal, P-Health, P-Employ
WHERE P-Personal.pid=P-Health.pid and
    P-Health.pid=P-Employ.pid and
    P-Personal.zipcode=145568 and
    P-Employ.salary > 10000 and
    P-Health.disease='diabetic'
```

Fig. 3. Audit Expression-2

3.1 Target Data View

The target data view defines the sensitive data which is in the audit scope. We denote this sensitive data as a set of data facts U and is obtained via predicates in the *WHERE* clause from the cross-product of tables in the *FROM* clause of the audit expression. We define the scheme of data facts U as the union of all attributes in the AUDIT and WHERE clause, and the tuple id attribute for each table present in the *FROM* clause of the audit expression. We denote the tuple-id attribute for a table T by tid_T. For example, target data facts U for the audit expressions given in Figures: 2, 3 and the relations shown in Tables: 1, 2, 3 would be as given in tables 4 and 5.

An auditing process involves analysis of user queries on the actual data contents accessed by them. On the other hand, a database state is updated many times due to insert, update and delete operations. Due to this change in database states, two similar queries q_{t_1} and q_{t_2} executed at different times may have different result set. Hence, due to possibility of many data versions existing for a given time interval, it becomes necessary to have mechanisms for data version specification to specify the intended target data view for auditing. We have identified requirements related to data versions

tid_{PP}	**name**	**age**	**address**
t11	Jane	25	A1
t13	Robert	29	A3
t14	Lucy	20	A4

Table 4. Target Data Facts U for Audit Expression 2

tid_{PP}	**pid**	**name**	**zipcode**	**address**	tid_{PH}	**disease**	tid_{PE}	**salary**
t12	p2	Reku	145568	A2	t22	diabetic	t32	20000
t14	p28	Lucy	145568	A4	t24	diabetic	t34	19000

Table 5. Target Data Facts U for Audit Expression 3

for a precise target view definition through an audit expression. The auditor may need to specify:

1. a set of data versions in a given time interval,
2. the current version of the data, and
3. a specific version of the data other than current data version.

We propose to use a *DATA-INTERVAL* clause in an audit expression. This clause would determine the set of target data versions. It may be noted that semantics of this clause are different from the clause *DURING* (explained later) discussed in [12]. The latter specifies the time interval for user queries which are to be audited. A *DATA-INTERVAL* clause has a pair of starting and ending time stamps (t_s, t_e). We use the keyword *now()* to denote the current time of the system.

As an example, the following expression

```
DATA-INTERVAL   1/5/2004:13-00-00 to now()
Audit           name, age, address From b-P-Personal Where  age < 30
```

would define a target data view with all the data versions from *1/5/2004:13-00-00* to *current system time*. In absence of this clause, we define the default interval as the current day interval, i.e., *current date:00-00-00* **to** *current system time stamp*. A specific data version is specified by giving the same time-stamp as the starting as well as ending time stamp. For example, the current database instance or version can be specified by using *now()* as t_s and t_e.

3.2 Suspicion Model

In query based auditing of privacy violations, the auditor specifies the data which is under disclosure review along with a notion of suspicion. While the earlier proposed audit expression models assumed a default single suspicion notion, we provide the facility to the administrator for specifying suspicion notion in the audit query. This increases the expressiveness of the audit expression model. The notion of suspicion defines *granules* of data such that access to any granule from the set by a batch of queries would label the batch as suspicious. The notion of suspicion is subjective in the real world. However, an auditor needs to specify it precisely in an audit expression. In our case, a suspicion notion defines a set of suspicion granules G defined through the target data view U of the audit expression such that if a batch of queries Q accesses any granule $o \in G$, Q is marked suspicious. We present a model for defining suspicion notion. The model comprises (i) a notion to specify the scheme for granules, (ii) the number of data facts in a granule, and (iii) a notion of accessibility of a granule.

The clauses which define a granule set G are:

1. *AUDIT* and *INDISPENSABLE* clauses: to specify the schemes of granules in G.
2. *THRESHOLD* clause: to define the number of tuples in each granule $o \in G$. For a given threshold value k, a scheme of a granule, and number of tuples n in U, there would be ${}^{n}C_{k}$ granules with that scheme.

The *AUDIT* and the *INDISPENSABLE* clauses together define the scheme for the granules whereas the *THRESHOLD* clause defines the number of tuples to be selected from U for each granule o. *INDISPENSABLE* clause defines whether tuple ids should be included or excluded in the granules scheme. Determining tuple ids that are to be included for each granule would depend upon the *partial scheme* in case of *INDISPENSABLE* value as *true*. By *partial scheme* of a granule we mean the scheme of granule defined by using only AUDIT clause.

The auditing attributes specified in an audit expression are usually (i) attributes which specify the subject notion whose information has been misused, e.g., an individual identifier or a user category and (ii) the sensitive attributes specifying the sensitive information associated with identifiers. In our opinion, following combinations of these attribute categories, identifiers and sensitive attributes, would be interesting to an administrator for auditing. The administrator would like to:

1. Specify a set of attributes as *optional* attributes such that a batch of queries would be marked suspicious if the batch has accessed *at least one* of the attributes. This requirement is clearly valid in the case of a set of sensitive attributes such that each attribute derives all other attributes in the set. For an example, in case of a set with *Bonus*, and *Salary* attributes and *Bonus* can be determined from *Salary* or vice-versa the case is true. By *optional* we mean ’one or more attributes’.
2. Specify a set of attributes as a combination of *mandatory* and *optional* attributes such that a batch of queries would be marked suspicious if the batch has accessed all the mandatory attributes and at least one attribute from the set of *optional* attributes. This requirement is valid in the case of a set of identifier attributes and a set of sensitive attributes such that each sensitive attribute derives all other sensitive attributes.
3. Specify a set of attributes as *mandatory* attributes such that a batch of queries would be marked suspicious if the batch accesses all the mandatory attributes. It is valid in the case of one sensitive attribute and all other as identifying attributes.

To specify all the above requirements, we use brackets $[]$ to specify a set of *optional* attributes, and parenthesis $()$ to specify mandatory attributes in the audit clause of an audit expression. For example, if an audit list has four attributes a,b,c,d and all attributes are sensitive such that access to any attribute would make a query batch suspicious, then this case can be specified as [*a*,*b*,*c*,*d*]. In order to specify attributes *a*,*b* as mandatory with attributes *c*,*d* as *optional*, the specification would be (*a*,*b*),[*c*,*d*] and would mark a batch of queries suspicious accessing attributes *a*,*b*,*c* or *a*,*b*,*d*. The specification (*a*,*b*,*c*,*d*) would make all four attributes mandatory and hence a query batch accessing all attributes would be marked as suspicious. We define a set of structural rules which describe the equivalence notions for different audit clause specifications in Table 6. Here in the rules specification, a capital letter denotes a set of attributes and a lower case letter denotes a single attribute.

No.	**Rule**	**Description**
1	$[a] = (a)$	An *optional* set containing a single element is equal to a mandatory set having the same element.
2	$(A)(B) = (A,B)$	A sequence of two mandatory sets yields a single mandatory set.
3	$(A,B) = (B,A)$ $[A,B] = [B,A]$	Set commutativity
4	$[a][b] = (a,b)$	Using rule-1 and rule-2
5	$[A][B] = [B][A]$ $(A)(B) = (B)(A)$ $(A)[B] = [B](A)$	Sequence commutativity
6	$[(A,B)] = (A,B)$ $([A,B]) = [A,B]$	Nesting
7	$(A,B)[c] = (A,B,c)$	Composition

Table 6. Audit Attributes Structural Rules

The number of data facts in each granule of set *G*, as discussed earlier, is decided by the *THRESHOLD* clause. We define the default value for THRESHOLD clause as 1 in case the administrator does not specify it. A special value *ALL* is also defined, which will include as many tuples in each granule as there are in the target data view *U* of an audit expression.

The notion of accessibility for each granule is decided from the presence or absence of special attribute tuple id, tid. If these attributes are present in granules, then a granule is treated as accessed by a query batch if all the tuples in the granules are *indispensable* for the query batch and the batch has accessed all these tuples. The indispensability notion is defined earlier (Definition 2). On the other hand, if tuple ids are not present in granules, i.e., *INDISPENSABLE* = *FALSE*, then the entire data in a granule would be treated as accessed by a batch of queries if the batch has accessed an information which contains tuples similar to the ones present in the granule. The presence of tuple id values in granules makes it necessary that the *WHERE* clause predicates of a query in the batch are consistent with the predicates in the *WHERE* clause of the audit expression to mark the batch as suspicious for that audit expression.

It can be seen that for a table U having k columns (assuming one column for tuple id attribute) and n records, $2^k.2^n - 1$ suspicion notions can be defined. We now show the expressability of our presented suspicion model by specifying the earlier discussed suspicion notions [12, 13, 17].

Notion of Perfect Privacy [17]. This notion of suspiciousness can be classified as the strongest notion of privacy proposed till now as it marks a query batch Q as suspicious if the batch Q has accessed data of any one of the cell in U. The granules set G for this notion would have each cell of U with tuple id attribute as its granules. For example, for the the audit expression in figure 4 the granules set G would be:

Granules Set:
G={*(t12,p2),(t22,p2),(t32,p2),(t12,145568),(t12,M),(t12,A2),(t12,Reku),(t22,W12), (t22,Nicholas),(t22,diabetic),(t22,drug1),(t32,E2),(t32,20000)*}

```
INDISPENSABLE = true
AUDIT [*] FROM P-Personal, P-Health, P-Employ
WHERE P-Personal.pid=P-Health.pid and
    P-Health.pid=P-Employ.pid and
    P-Personal.zipcode=‘‘145568" and
    P-Employ.salary > 10000 and
    P-Health.disease=‘‘diabetic" and
    p-Personal.name=’Reku’;
```

Fig. 4. Audit Expression for Perfect Privacy

The perfect privacy always considers all the columns of tables specified in the ’from’ clause of audit query in the suspicion granules.

Notion of Weak Syntactic Suspicion [13]. This suspicion notion marks a batch of queries Q suspicious if Q accesses any column specified in audit expression and constraints of batch Q are not in conflict with constraints of the audit expression, i.e., there is some indispensable tuple between the batch Q and audit expression A. Therefore, the granules in this case would be each pair of values of the column value of one of the columns specified in the audit list and the tuple id tid of the respective row in the table. For example, if we use the audit expression given in figure 5 then the granules set G would be:

Granules Set:
G={*(t12,p2),(t12,145568),(t12,Reku),(t12,A2),(t14,p28), (t14,145568),(t14,Lucy), (t14,A4),(t22,diabetic),(t24,diabetic), (t32),(t32,20000),(t34,19000),(t22,p2),(t32,p2), (t24,p28), (t34,p28)*}

```
INDISPENSABLE = true
AUDIT [name,disease,address,P-Personal.pid,
P-Health.pid, P-Employ.pid, zipcode, salary]
FROM P-Personal, P-Health, P-Employ
WHERE P-Personal.pid=P-Health.pid and
    P-Health.pid=P-Employ.pid and
    P-Personal.zipcode=145568" and
    P-Employ.salary > 10000 and
    P-Health.disease=diabetic"
```

Fig. 5. Audit Expression for Weak Syntactic Suspicion Notion

Notion of Indispensable Tuple [12] or Strong Semantic Suspicion [13] This suspicion notion is stronger than the earlier explained notions. It marks a batch of queries *Q* as suspicious if the query batch has accessed all the columns specified in the audit list and the tuple is indispensable. Therefore, for the audit expression from figure 6, the granules set G would be:

Granules Set:
G={*(t12,t22,Reku,diabetic,A2),(t14,t24,Lucy,diabetic,A4)*}

```
INDISPENSABLE = true
AUDIT (name,disease,address)
FROM P-Personal, P-Health, P-Employ
WHERE P-Personal.pid=P-Health.pid and
    P-Health.pid=P-Employ.pid and
    P-Personal.zipcode=‘‘145568" and
    P-Employ.salary > 10000 and
    P-Health.disease=‘‘diabetic"
```

Fig. 6. Audit Expression for Semantic Suspiciousness

3.3 Limiting Parameters

In a privacy enforced information system, any user access to information is filtered through the privacy policy. Therefore, the authorization parameters given in the privacy policy which allow access to the *target data view* can be specified in an audit expression. These parameters are usually *User-id*, *Purpose-id*, *Role-id* and can be specified in a negative way; user accesses with these parameters will not be considered for the auditing, or in positive way; user accesses with these parameters are considered for the auditing. In case of a conflict between the authorization parameters in both clauses, i.e., the positive clause allows it and the negative clause denies it, we give precedence to negative clause and the accesses will not be audited. There is no specific reason to give precedence to negative clause here except to resolve the conflict. We would use the following clauses for specification of privacy policy specific parameters.

1. **Neg-Role-Purpose** $\{(r, pr)|(r, -)|(-, pr)\}^*$
2. **Pos-Role-Purpose** $\{(r, pr)|(r, -)|(-, pr)\}^*$
3. **Neg-User-Identity** $\{u - id\}^*$
4. **Pos-User-Identity** $\{u - id\}^*$

Neg-Role-Purpose is a list of ordered pairs of role and purpose, the semantics of which is to not consider the user accesses with these parameters for auditing. The (r, pr) ordered pair in this clause indicates to remove all user accesses having r as a role and pr as a purpose annotation in the User Accesses Log from consideration. The $(r, -)$ ordered pair removes all the accesses having r as their role (- denotes any purpose). Similarly, $(-, pr)$ removes all accesses having pr as the access purpose. If the administrator has the information of role and purpose through which a violation has occurred (i.e., positive aspect) then that can be specified in *Pos-Role-Purpose* clause. If information about user identities is known, it is specified similarly in the *Pos-User-Identity* and *Neg-User-Identity* clauses.

Other limiting information is the time interval for user accesses. We use the *DURING* clause proposed in [12]. The user accesses made to database in the ’DURING’ interval are to be audited.

3.4 Final Audit Expression

We now define all the clauses used to specify an audit expression (Figure 7). The clauses having default values are optional and need not to be specified. The limiting parameters filter user accesses from auditing.

The proposed audit expression model facilitate to specify the intended target data view even in presence of data versions. Hence, the model helps to specify precise information and prevents ambiguity. The model is more expressive as it allows to specify suspicion notion along with the version related information. It is flexible as it does not require to specify all the clauses, i.e., some clauses are optional.

DURING clause filters the accesses which are not in the specified interval, whereas *DATA-INTERVAL* helps in determining the target data view. All the satisfying tuples are collected from each specified database state using *DATA-INTERVAL*. The methodology to get database instance as proposed in [12] can be used for this purpose.

It could be seen that the proposed expression syntax and semantics of audit expression is capable of expressing all the identified aspects in the expression. Thus the presented audit expression model is more expressible and fulfills the need of an auditor for the task of determining relevant suspicious queries.

4 Conclusion and Future Work

In query based auditing, the administrator enters an audit expression to identify the suspicious accesses for a given privacy violation. We have presented audit expression

```
Neg-Role-Purpose   {(r,pr)|(r,-)|(-,pr)}*
!------(default is to consider all user accesses)
Pos-Role-Purpose   {(r,pr)|(r,-)|(-,pr)}*
!------(default is to consider all user accesses)
Neg-User-Identity  {u-id}*
!------(default is to consider all user accesses)
Pos-User-Identity  {u-id}*
!------(default is to consider all user accesses)
DURING              timestamp 1 to timestamp 2
!------(default is current day)
DATA-INTERVAL       timestamp1 to timestamp2
!------(default is current day)
THRESHOLD N
!------(default is 1)
INDISPENSABLE      true | false
!------(default is true)
AUDIT               attribute list
FROM                table names
WHERE               conditional expression
```

Fig. 7. Proposed Audit Expression Syntax

model for these expressions. The model unifies earlier proposed audit expressions and consists of target data view, suspicion notions, and limiting parameters. We have given mechanism to define the target data view even in presence of database updates. It is also shown how the earlier notions can be specified using our presented suspicion model. The limiting parameters, one of the constituents of the audit expression model, identifies the context information which can be specified in an auditing expression for privacy violation detection. We use privacy policy parameters and a time duration for this purpose. The proposed audit expression would help the auditor to specify an audit expression to retrieve relevant and intended suspicious queries.

As a future work, it would be interesting to see for what suspicion notions static determination of a query batch suspiciousness for an audit expression is decidable. Further, future work includes designing efficient algorithms to map an audit expression to a set of suspicious batch of queries for a given database instance. In case of on line auditing, there is a need to determine the suspicion rank, closeness value, of a queries batch for a given set of audit expressions. Therefore, an interesting task would be to use the presented audit expression model for computing the degree of suspiciousness for user queries on line.

5 Acknowledgments

The authors acknowledge with thanks support from the projects ”Design and development of Malafide intention based Privacy Violation Detection System” sponsored by Department of Information Technology, and ”Advanced Information Systems Security Laboratory”.

References

1. : OASIS, eXtensible Access Control Markup Language (XACML) TC. http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml
2. Ashley, P., Hada, S., Karjoth, G., Powers, C., Schunter, M.: Enterprise Privacy Authorization Language (EPAL 1.1). http://www.zurich.ibm.com/security/enterprise-privacy/epal (2003) IBM Research Report.
3. Bhattacharya, J., Gupta, S.K.: Privacy Broker for Enforcing Privacy Policies in Databases. In: Proceedings of Fifth international conference on knowledge based computer systems, Hyderabad, India (2004)
4. Agrawal, R., Kiernan, J., Srikant, R., Xu, Y.: Hippocratic Databases. In: Proceedings of the 28th International Conference on VLDB, Hong Kong, China (2002) 143–154
5. Rosencrance, L.: Toysrus.com faces online privacy inquiry. http://archives.cnn.com/2000/TECH/computing/12/14/toysrus.privacy.inquiry.idg/toysrus.privacy.inquiry.html
6. : Associated Press: Fliers File Suit Against Jetblue. http://www.wired.com/politics/security/news/2003/09/60551 (Sep 23 2003)
7. Barse, E.L.: Logging For Intrusion And Fraud Detection. PhD Thesis, ISBN 91-7291-484-X Technical Report no.28D ISSN 1651-4971, School of Computer Science and Engineering, Chalmers University of Technology (2004)
8. Bruno, J.B.: Security Breach Could Expose 40M to Fraud. http://www.freerepublic.com/focus/f-news/1425334/posts (June 18 2005)
9. Teasley, B.: Does Your Privacy Policy Mean Anything? http://www.clickz.com/experts/crm/analyze_data/article.php (January 11 2005)
10. Goyal, V., Gupta, S.K., Saxena, S., Chawala, S., Gupta, A.: Query Rewriting for Detection of Privacy Violation through Inferencing. In: International Conference on Privacy, Security and Trust (PST06), supported by ACM SIGSAC, Markham, Ontario, Canada (Oct 30 - Nov 1, 2006) 233–243
11. Gupta, S.K., Goyal, V., Patra, B., Dubey, S., Gupta, A.: Design and Development of Malafide Intension Based Privacy Violation Detection System (An Ongoing Research Report). In: International Conference on Information Systems Security (ICISS), Kolkata, India, Lecture Notes in Computer Science-4332, Springer Verlag (2006) 369–372
12. Agrawal, R., Bayardo, R., Faloutsos, C., Kiernan, J., Rantzau, R., Srikant, R.: Auditing compliance with a Hippocratic database. In: VLDB ’04: Proceedings of the Thirtieth international conference on Very large data bases, VLDB Endowment (2004) 516–527
13. Motwani, R., Nabar, S., Thomas, D.: Auditing a Batch of SQL Queries. In: IEEE 23rd International Conference on Data Engineering Workshop. (2007) 186–191
14. Böttcher, S., Steinmetz, R.: Detecting Privacy Violations in Sensitive XML Databases. In: 2nd VLDB Workshop on Secure Data Management Secure Data Management-SDM 2005,, Trondheim, Norway, Lecture Notes in Computer Science-3674 (2005) 143–154
15. Gupta, S.K., Goyal, V., Gupta, A.: Malafide Intension Based Detection of Violation in Privacy. In: International Conference on Information Systems Security (ICISS), Kolkata, India, Lecture Notes in Computer Science-4332, Springer Verlag (2006) 365–368
16. Machanavajjhala, A., Gehrke, J.: On the Efficiency of Checking Perfect Privacy. In: PODS ’06: Proceedings of the twenty-fifth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems, New York, NY, USA, ACM Press (2006) 163–172
17. Miklau, G., Suciu, D.: A Formal Analysis of Information Disclosure in Data Exchange. J. Comput. Syst. Sci. **73**(3) (2007) 507–534
18. Reiss, S.P.: Security in databases: A combinatorial study. J. ACM **26**(1) (1979) 45–57