Auditing SQL Queries

Rajeev Motwani [#1], Shubha U. Nabar [#2], Dilys Thomas [*3]

[#] *Computer Science Department, Stanford University*
[1] rajeev@cs.stanford.edu
[2] sunabar@cs.stanford.edu

[*] *Google Inc.*
[3] dilysthomas@gmail.com

***Abstract*— We study the problem of auditing a batch of SQL queries: Given a forbidden view of a database that should have been kept confidential, a batch of queries that were posed over this database and answered, and a definition of suspiciousness, determine if the query batch is suspicious with respect to the forbidden view. We consider several notions of suspiciousness that span a spectrum both in terms of their disclosure detection guarantees and the tractability of auditing under them for different classes of queries. We identify a particular notion of suspiciousness, *weak syntactic suspiciousness*, that allows for an efficient auditor for a large class of conjunctive queries. The auditor can be used together with a specific set of forbidden views to detect disclosures of the association between individuals and their private attributes. Further it can also be used to prevent disclosures by auditing queries on the fly in an online setting. Finally, we tie in our work with recent research on query auditing and access control and relate the above definitions of suspiciousness to the notion of unconditional validity of a query introduced in database access control literature.**

I. Introduction

Auditing is the process of inspecting past actions to determine whether they were in conformance with official policies. In the context of database systems with data disclosure policies, auditing queries is the process of inspecting queries that have been answered in the past and determining whether these answers could have been pieced together by a user to infer confidential information.

More formally, given a set of forbidden views $\mathcal{V} = \{V_1, \ldots, V_k\}$ of a database that must be kept confidential, a batch of queries, $\mathcal{Q} = \{Q_1, \ldots, Q_m\}$, that have been posed over this database and answered, and a system-defined notion of suspiciousness S, the task of an auditor is to determine whether $\mathcal{Q}$ is suspicious with respect to $\mathcal{V}$. In this paper, we focus on the case of a single forbidden view.

The need for an audit mechanism in database management systems is clear. For example, an individual on receiving targeted health advertisements might suspect his health-care provider of having leaked private information from his medical records. If the provider's privacy policy stipulates that it does not release patient data to external parties, it would be in the best interests of the provider to be able to demonstrate compliance with this policy.

Auditing could also be studied in an *online* context — as new queries are posed by a user, when should queries be denied to prevent information leaks about a forbidden view. We will discuss this briefly in Section VI, but for the most part will focus on retro-active auditing highlighted above.

Prior work in this area includes [2] where a definition of suspiciousness was introduced for a single query on its own and [8] where the problem of ensuring *perfect privacy* of the forbidden view was considered, and a resulting definition of suspiciousness was developed. In this paper we consider other notions of suspiciousness that lie in between those of [2] and [8] both in terms of their disclosure detection guarantees and the tractability of auditing under them.

In general no single query on its own may be the cause of a disclosure, therefore in Section III we first extend the definition of suspiciousness in [2] to multiple queries. For reasons that will become evident later, we call this *semantic suspiciousness*. We provide a simple *semantic auditor* for auditing a batch of select-project-join (SPJ) queries under this definition. Semantic suspiciousness, however, doesn't guarantee detection of certain basic forms of disclosure of the forbidden view. Further, the semantic auditor requires actual execution of the queries in the query batch against the database.

A natural question to ask is whether this could be avoided. To this end in Section IV, we further extend semantic suspiciousness to a database instance independent notion of *strong syntactic suspiciousness*. The hope is that by so doing, suspiciousness of a query batch could be determined simply by looking at the structure of the queries themselves without having to execute them against the database. Unfortunately we show here that this is in fact NP-hard to achieve even when we restrict ourselves to the class of conjunctive select-project (SP) queries without inequalities.

In Section IV-B, we consider *weak syntactic suspiciousness* — a relaxation of the above strong definition. This relaxation provides many advantages over semantic and strong syntactic suspiciousness. First, it has better disclosure detection guarantees than either of the two. Second, auditing a large class of conjunctive queries now becomes feasible with a simple algorithm that is polynomial in just the size of the input query expressions. Further, as discussed in Section VI, since the definition does not depend on the actual underlying database, a weak syntactic auditor could also be used to audit queries on the fly in an online setting.

While a weak syntactic auditor may not provide as strong a disclosure detection or prevention guarantee as the notion of perfect privacy introduced in [8] (which would label many

seemingly innocuous queries as suspicious), we show in Section V that in conjunction with a specific set of forbidden views, it can nevertheless provide a very strong guarantee that would be sufficient in many practical scenarios when all we wish to protect is the associations between identifiers and private attributes. At the same time, the auditor is simple and works for a very large class of queries.

In Sections VI and VII we summarize the relationships between the different auditors above and also draw a connection to another database mechanism for controlling access to data — namely determining query validity studied in [9], [14], [15], [13]. The connection is interesting in that ideally a database system would have both an auditing component as well as an access control component and it is important to understand the interaction between the two.

Assumptions: The auditors that we design in the upcoming sections are able to *exactly* determine query suspiciousness only when they operate on *duplicate-preserving* SPJ queries and views where the `SELECT` clause does not contain `distinct` and multi-set semantics is used in projection. We drop the duplicate-preserving qualifier from here on, but it is assumed as a default. Our suspiciousness definitions, however, are independent of query semantics, and at the end of Section IV-B, we briefly consider the guarantees of our auditors when queries do have a set semantics.

We allow queries with self-joins, but only consider forbidden views with no self-joins. For ease of exposition, we only show our results here for queries without self-joins. Unless otherwise stated, the results can be easily extended.

Furthermore, we make some assumptions about the domain. Let $\mathcal{C}$ be the set of all columns (attributes) across all tables in the database. For any $C \in \mathcal{C}$, let $dom(C)$ denote the domain of column C and let it be of size at least 2. Let $dom(T)$ denote the set of possible tuples for a table T that can be obtained by assigning constants from $dom(C)$ for each column, C, in T. Thus the value that an attribute of a tuple can take is not constrained by the values of other attributes. A database instance is then a multi-set of tuples in $dom(T)$ for every possible T that occurs in the database. We next briefly review relevant past work in auditing.

II. Preliminaries and Related Work

A. Auditing Aggregate Queries

The problem of auditing queries has been extensively studied in the context of statistical databases [5], [12], [3], [1], [4], [6], [11]. Statistical databases allow users to retrieve only aggregate statistics over subsets of its data. In this paper we consider only SPJ queries and our work is orthogonal to the body of work on statistical databases.

B. Perfect Privacy

In [8], [7] the authors consider the problem of ensuring "perfect privacy": as a database system reveals various views of its data, does it disclose any information *at all* about a secret view that was required to be kept confidential. This work can also be cast in the auditing framework proposed in this paper — the secret view corresponds to the forbidden view in our scenario, the views of the data that were released correspond to queries that were answered and the privacy requirement results in a definition of suspiciousness for identifying queries that violated the requirement.

Perfect privacy: Let D be a distribution according to which tuples of the database are drawn. A set of released views, $\mathcal{Q}$, is said to respect perfect privacy of a forbidden view, $\mathcal{V}$, if:

$$Pr_D\{\mathcal{V} = V\} = Pr_D\{\mathcal{V} = V|\mathcal{Q}\}$$

$Pr_D\{\mathcal{V}\}$ is the prior distribution over the forbidden view and $Pr_D\{\mathcal{V}\,|\mathcal{Q}\}$ is the posterior distribution given the released views. Perfect privacy is very strict and results in the following criterion for query suspiciousness (referred to in this paper as *perfect privacy suspiciousness*) for distributions D where each tuple is included in the database independently with some probability and queries and views follow set semantics:

Definition 1: (Critical Tuple) A tuple t belonging to the domain of all possible tuples in the database, is critical for a query Q, if there exists a possible database instance I for which $Q(I - \{t\}) \neq Q(I)$, i.e., if there exists an instance in which dropping t makes a difference to the result of Q.

Definition 2: (Perfect Privacy Suspiciousness) A query batch $\mathcal{Q}$ is suspicious with respect to a view $\mathcal{V}$ iff some query $Q \in \mathcal{Q}$ is suspicious. Furthermore Q is suspicious with respect to to $\mathcal{V}$ iff there exists some critical tuple common to both $\mathcal{V}$ and Q.

Disclosure detection guarantee: When queries are audited under perfect privacy suspiciousness, an unsuspicious query batch is guaranteed to respect perfect privacy of the forbidden view, whereas a suspicious query batch violates it. Even though this guarantee is shown to hold only under set semantics and for data distributions where a database instance consists of a set (not a multi-set) of tuples drawn independently of one another, we still treat it as a candidate notion of suspiciousness for duplicate preserving queries. We do not make any claims of its guarantees in this setting. Perfect privacy suspiciousness is very strict, marking many seemingly innocuous queries as suspicious. For example:

Consider a hospital database containing the names and phone numbers of patients and we wish to keep secret all phone numbers in the database, i.e., the secret view is $\pi_{phone}(Patients)$. A query, $\pi_{name}(Patients)$, asking for the names of all the patients in the hospital would be considered suspicious with respect to the secret view even though not a single phone number was revealed by this query. This is because every possible tuple in $dom(Patients)$ is critical to both the query and the secret view. The idea is that by revealing information about the size of the database, the query revealed some information about the phone numbers column and is considered suspicious.

In [7], the authors identify subclasses of conjunctive queries without inequalities ($\leq, <, \geq, >, \neq$), following set semantics, for which checking for perfect privacy suspiciousness

is tractable. But users often pose queries with a multi-set semantics. Is checking perfect privacy suspiciousness simpler in this scenario? There isn't an immediately obvious answer (see end of Section IV-B). Partly due to this, and partly because of the strictness of the information disclosure requirements of perfect privacy, we search for other suspiciousness definitions.

C. Auditing SQL Queries

In [2] the authors study the problem of determining if any single query in the query log accessed private information. The data subject to a disclosure review is specified very simply through an *audit expression* that closely resembles a SQL query. In the remainder of this paper, we too use audit expressions to specify forbidden views:

```
AUDIT audit list FROM table list
WHERE condition list
```

The audit expression is viewed as an SPJ query specifying the forbidden view. The sensitive information is in particular the `audit list` columns of the tuples of the forbidden view. The tuples of interest are identified from the cross-product of tables in the `FROM` clause via predicates in the `WHERE` clause. The definition of suspiciousness considered in [2] asks for all queries whose `WHERE` clause predicates were satisfied by *any* of these forbidden tuples and that accessed *all* of the `audit list` columns for that tuple. Formally,

Let $Q = \pi_{C_Q}(\sigma_{P_Q}(\mathcal{R}))$ be an SPJ query and $A = \pi_{C_A}(\sigma_{P_A}(\mathcal{S}))$ be an audit expression. Here $\mathcal{R}$ and $\mathcal{S}$ are the cross-product of all tables in the `FROM` clauses of Q and A. $C_{Q(resp.\ A)}$ are the columns that are projected out in Q (resp. A) and $P_{Q(resp.\ A)}$ are the predicates of Q (resp. A). C^*_Q are all the column names that appear anywhere in Q.

Definition 3: (Candidate Query) Q is a candidate query with respect to A, if Q accesses all the `audit list` columns: $C^*_Q \supseteq C_A$.

Definition 4: (Indispensable Tuple) Let $\mathcal{T}$ be the cross-product of some subset of tables in Q and let $\mathcal{R}'$ be the cross product of the remaining. Thus $\mathcal{T} \times \mathcal{R}' = \mathcal{R}$. A tuple $t \in \mathcal{T}$ is indispensable to Q if the presence or absence of t makes a difference to the result of Q, i.e., $\pi_{C_Q}(\sigma_{P_Q}(\mathcal{T} \times \mathcal{R}')) \neq \pi_{C_Q}(\sigma_{P_Q}(\mathcal{T} - \{t\} \times \mathcal{R}'))$.

Definition 5: (Suspicious Query) Let $\mathcal{T}$ be the cross product of tables common to A and a candidate query Q. Q is suspicious for A if they share an indispensable tuple $t \in \mathcal{T}$.

The idea is that an indispensable tuple for A is one of the forbidden tuples being subjected to a disclosure review. So if any one of these tuples is also indispensable for Q, then Q revealed some information about the forbidden tuple while accessing all of its `audit list` columns in the process. Therefore Q potentially revealed information about the `audit list` columns themselves for this forbidden tuple. For example, consider the audit expression:

```
AUDIT p.disease FROM Patients p
WHERE p.zipcode = 94305
```

Under the suspiciousness definition, this expression asks for queries that returned information about any patient living in 94305 and simultaneously accessed his disease column. Now consider the query:

```
SELECT p.zipcode FROM Patients p
WHERE p.disease = 'diabetes'
```

If any patient who has diabetes lives in 94305, this query will be considered suspicious. It would not, however, be suspicious with respect to the following view:

```
AUDIT p.zipcode FROM Patients p
WHERE p.disease = 'hypertension'
```

This is because this audit expression is only interested in checking if the zipcode of any patient with hypertension was revealed. For the class of duplicate-preserving SPJ queries, the condition of sharing an indispensable tuple translates to the following: Let $\mathcal{T}$ be the cross-product of tables common to A and a candidate query Q and $\mathcal{S}'$ and $\mathcal{R}'$ be the cross-product of the remaining tables in A and Q respectively. Q is suspicious iff $\sigma_{P_A}(\sigma_{P_Q}(\mathcal{T} \times \mathcal{R}' \times \mathcal{S}')) \neq \emptyset$. So now the audit process is simple: for every candidate query, Q, in the query log, if the result of running $\sigma_{P_A}(\sigma_{P_Q}(\mathcal{T} \times \mathcal{R}' \times \mathcal{S}'))$ is non-empty, then Q is suspicious.

Disclosure detection guarantee: The guarantees and weaknesses of this suspiciousness definition and auditor are not discussed in [2]. Informally, it detects queries whose results contain tuples corresponding to forbidden tuples and that on their own could have leaked information about all the `audit list` columns of such a forbidden tuple to an adversary with no background knowledge of correlations between tuples or between attributes of the same tuple. We show that it may not however catch a query that could have leaked information about the `audit list` columns of a forbidden tuple because of the absence of a corresponding tuple in its results. Consider the following audit expression and query:

```
AUDIT P.disease FROM Patients p
WHERE p.name = 'Alice'
```

```
SELECT P.name FROM Patients p
WHERE p.disease = 'cancer'
```

If Alice does not have cancer, the query would be considered safe, although it did reveal that either Alice was not in the database, or she did not have cancer. Such negative disclosures are not detected. This is not true for the syntactic notions of suspiciousness that we propose.

Indispensability and criticality: Note that if a tuple t is indispensable to a query, then a portion of the tuple t' that comes from any single base table is a critical tuple for the query. This is because in a duplicate-preserving query, there is a one-to-one correspondence between the number of input tuples that satisfy its `WHERE` clause predicates and the number of output tuples. Removing t' from the current database instance reduces the number of satisfying input tuples and thus changes the result of the query.

We will now introduce our new notions of suspiciousness, design auditors for them, and discuss their disclosure detection guarantees. Please see a technical report at [10] for discussions and proofs of some of the results that have been omitted here for lack of space.

III. Semantic Auditing

The work that we first chose to build on was that of [2] since the notion of suspiciousness here is more relaxed than prefect privacy suspiciousness and allows for simpler auditors. The authors however, only considered detecting queries that on their own accessed all the forbidden columns of one or more of the tuples of a forbidden view. In general, no one query in isolation may have accessed all the `audit list` columns of a forbidden tuple, though a few queries together could have For example, the audit expression might be

```
AUDIT p.name, p.disease FROM Patients p
WHERE p.zipcode = 94305
```

Here the data that needs to be kept secret is the association between names and diseases of patients living in the zipcode 94305. Now consider the SQL queries:

```
SELECT p.zipcode FROM Patients p
WHERE p.disease = 'diabetes'

SELECT p.name FROM Patients p
WHERE p.zipcode = 94305
```

Neither query on its own could have revealed the name-disease association of anyone living in 94305 to an attacker with no background knowledge of people living in 94305. However the combination of the queries could have. For instance, if there was only one patient in 94305 and he had diabetes, then these queries revealed his name-disease association. There are subtle ways in which the results of queries could be combined to reveal information and in the notion of suspiciousness we introduce here, informally, we seek to capture disclosures of the `audit list` columns that could be caused by an attacker "joining" query results that contain tuples corresponding to forbidden tuples.

We assume that attackers are powerful and know exactly which tuples in the results of different queries come from the same individual and can be "joined". The resulting suspiciousness definition is conservative: it may label query batches as suspicious when a user could not have easily mapped result tuples to a particular individual in reality.

Since this form of suspiciousness derives meaning from a particular world view, i.e., the current database state, we call it *semantic suspiciousness*. This is in contrast to *syntactic suspiciousness* defined later, where suspiciousness of a batch depends entirely on the structure of the queries in the batch, not on the actual database state.

Definition 6: (Semantically Suspicious Query Batch) A query batch $\mathcal{Q}$ is semantically suspicious with respect to an audit expression A if there is some subset of queries $\mathcal{Q}' \subseteq \mathcal{Q}$ and some tuple $t \in \mathcal{T}$ (the cross-product of tables common to A and every query $Q \in \mathcal{Q}'$) such that (1) t is indispensable to both A and every $Q \in \mathcal{Q}'$ and (2) the queries in $\mathcal{Q}'$ together access all the `audit list` columns of A.

Semantic suspiciousness is a natural extension of the definition in [2] and we construct an auditor for it.

Theorem 1: Auditor 1 determines exactly if an SPJ query batch $\mathcal{Q}$ is semantically suspicious with respect to audit expression A.

Algorithm 1 Semantic Auditor

```
1: // D=cross-product of tables occurring in A or some Q ∈ Q
2: compute V = π_*(σ_{P_A}(D)) on the given database instance
3: for every query Q ∈ Q
4:     for every tuple t ∈ V
5:         if t satisfies the WHERE clause predicates of Q
6:             mark all columns of t accessed by Q
7: if all audit list columns of some t ∈ V are marked
8:     return suspicious.
9: else return unsuspicious.
```

To determine exactly the subset of queries involved in the disclosure, the auditor can maintain with each marked cell, the set of queries that accessed it. The cross-product of all the query sets for each cell of each disclosed tuple gives all the query subsets $\mathcal{Q}'$ that could have caused the disclosure.

We omit a proof due to space constraints. The idea is that if a tuple $t \in V$ satisfies the predicates of a set of queries $\mathcal{Q}'$, then the part of t, t', that comes from the cross-product of tables common to all queries in $\mathcal{Q}'$ and A is indispensable to every query in $\mathcal{Q}'$ and A. This is because the queries are duplicate-preserving: every input tuple that satisfies their predicates results in a corresponding output tuple. So removing t' causes a difference to all the query results.

Note that as in [2], we require the query batch to access *all* the `audit list` columns in order to be even considered for suspiciousness testing. This is because informally, we would ideally like to determine "full disclosure" of a forbidden tuple, i.e., could any combination of queries in the query batch have revealed all the `audit list` columns of a forbidden tuple to an adversary. This is motivated by findings such as in [16] where attributes such as date-of-birth, gender or zipcode on their own are not selective, however in conjunction can uniquely identify individuals. In such a case, revealing the diseases of all females in a zipcode is likely to be harmless, however revealing this in conjunction with dates-of-birth could cause significant privacy breaches.

We discuss the disclosure detection abilities of a semantic auditor in more detail in Section V. Briefly, just like the auditor of [2], it does not catch query batches that could have caused disclosure of the `audit list` columns due to the absence of forbidden tuples in query results. In fact the problem of negative disclosures becomes even more amplified for multiple queries since many negative disclosures together could add up to a lot of positive disclosure. It is partly due to this that we study other definitions of suspiciousness. Another reason is that executing every single query in $\mathcal{Q}$ against the forbidden view could be an expensive operation if the query log and the forbidden view are large. In the following section we consider

steps towards eliminating these drawbacks.

IV. Syntactic Auditing

We define notions of suspiciousness that are independent of the database instance. The hope is that by eliminating dependence on underlying data, suspiciousness can be determined from just the structure of the queries. As an interesting side effect, syntactic auditors can catch negative disclosures unlike semantic auditors. We give formal guarantees for certain kinds of forbidden views in Section V.

A. Strong Syntactic Suspiciousness

Definition 7: (Strong Syntactic Suspiciousness of a Query Batch) A query batch is strongly syntactically suspicious with respect to an audit expression if there is some database instance for which it is semantically suspicious.

Note that the current database tables form an instantiation. Therefore if a query batch is semantically suspicious with respect to an audit expression, it is also strongly syntactically suspicious. A strong syntactic auditor would thus be more conservative than a semantic auditor.

Unlike semantic auditing, the hope is that checking for strong syntactic suspiciousness would require the auditor to only analyze the structure of query expressions in the batch, not the answers to the queries on the actual database since suspiciousness is independent of the underlying database. Our result here is negative, even if we restrict ourselves to just the class of conjunctive select-project queries without inequalities.

Theorem 2: Testing exactly if a batch of conjunctive SP queries without inequalities is strongly syntactically suspicious with respect to an audit expression is NP-hard in the size of the query expressions.

Weakening this notion of suspiciousness, however, lets us make positive claims, while providing auditors that would only be more strict, i.e., all strongly suspicious query batches would also be classified as weakly suspicious.

B. Weak Syntactic Suspiciousness

Definition 8: (Weak Syntactic Suspiciousness of a Query Batch) A batch of 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 instantiation of database tables I such that (1) a tuple $t \in \mathcal{T}$ (the cross-product of tables common to every $Q \in \mathcal{Q}'$ and A) is indispensable to every $Q \in \mathcal{Q}'$ and A in the context of I and (2) the queries in $\mathcal{Q}'$ together access *at least one* of the `audit list` columns.

Note that to arrive at this definition, we simply weakened condition (2) in the definition of strong syntactic suspiciousness and thus a query batch that is strongly syntactically suspicious will also be weakly syntactically suspicious.

At this point, the definition may not seem all that different from perfect privacy suspiciousness, but as we shall see, the difference between indispensability and criticality enables us to design a simple weak syntactic auditor for a large class of queries and that cannot be used to detect perfect privacy suspiciousness. Note also that the weakening of condition (2) gives us the following decomposability result.

Theorem 3: An SPJ query batch $\mathcal{Q}$ is weakly syntactically suspicious with respect to an audit expression, A, iff some $Q \in \mathcal{Q}$ is weakly syntactically suspicious with respect to A.

Intuitively this is because of the shift in focus from trying to detect "full disclosures" of the `audit list` columns of forbidden tuples to trying to detect "partial disclosures". Suspiciousness of a batch can now be tested by testing suspiciousness of each query on its own (as is true for perfect privacy suspiciousness). Focusing on partial disclosures gets us away from the complex issues of how queries could be combined to reveal information about *all* the `audit list` columns because for a partial disclosure to occur, some query should have disclosed information about at least one of the columns to begin with. We formalize the disclosure detection guarantees of a weak syntactic auditor for a specific set of forbidden views in Section V, however it is clear that the following relationship holds:

Theorem 4: Perfect Privacy Suspiciousness $\geq$ Weak Syntactic Suspiciousness $\geq$ Strong Syntactic Suspiciousness $\geq$ Semantic Suspiciousness.

Here '$A \geq B$' means that definition A is stricter than definition B and therefore all query batches considered suspicious for a given audit expression by B would also be considered suspicious by A.

Despite Theorem 3, testing strong syntactic suspiciousness of an arbitrary SPJ query is still NP-hard in the size of the query expression. The proof is a straightforward reduction from 3-SAT.

Theorem 5: Testing weak syntactic suspiciousness of an arbitrary SPJ query batch with respect to an audit expression is NP-hard in the size of the query expressions.

A natural question is whether there are classes of queries for which there exist efficient weak syntactic auditors. We restrict ourselves to conjunctive SPJ queries and views that do not contain $\neq$ predicates between attributes in their `WHERE` clauses and show a positive result here. Note that we do allow $\neq$ predicates between attributes and constants.

Theorem 6: Let $\mathcal{Q}$ be a conjunctive SPJ query batch and $A = \pi_{C_A}(\sigma_{P_A}(S))$ be a conjunctive SPJ audit expression. If A and $\mathcal{Q}$ have no $\neq$ predicates between any two attributes in their `WHERE` clauses, then Auditor 2 can determine exactly if $\mathcal{Q}$ is weakly syntactically suspicious with respect to A in time polynomial in the size of the query expressions.

First, due to Theorem 3 it suffices to check suspiciousness of each query in the batch individually. If any one query is suspicious, we can mark the entire batch as suspicious. Second, we only check suspiciousness of queries that access at least one of the `audit list` columns of A. Suspiciousness of such a candidate query Q is determined by testing if P_A and P_Q are compatible, i.e., could any tuple in $dom(\mathcal{T} \times \mathcal{R}' \times \mathcal{S}')$ satisfy both P_A and P_Q. The query is weakly syntactically

Algorithm 2 Skeleton of a Weak Syntactic Auditor
(for conjunctive queries and views, step 3 can be determined in time polynomial in the size of the query expressions)

1: **for** every query $Q = \pi_{C_Q}(\sigma_{P_Q}(R)) \in \mathcal{Q}$
2: **if** $C^*_Q \supseteq C_A$
3: **if** $\exists\ t \in\ dom(\mathcal{T} \times \mathcal{R}' \times \mathcal{S}')$ such that $P_A(t) = 1$ and $P_Q(t) = 1$
4: **return** suspicious
5: **return** unsuspicious
6: //$\mathcal{T}$ = cross-product of tables common to Q and A
7: //$\mathcal{R}'$, $\mathcal{S}'$ = cross-products of remaining tables in Q and A

suspicious iff such a tuple exists. This is because Q and A are duplicate-preserving, ensuring a one-to-one correspondence between output tuples and input tuples that satisfy their `WHERE` clause predicates. So if a $t \in \mathcal{T} \times \mathcal{R}' \times \mathcal{S}'$ satisfies P_Q and P_A, the portion of t that comes from $\mathcal{T}$ will be indispensable to both Q and A in an instance containing that tuple. Likewise, if a $t \in \mathcal{T}$ is indispensable to both Q and A in an instance where $r' \in \mathcal{R}'$ and $s' \in \mathcal{S}'$, then $t \times r' \times s'$ will satisfy P_Q and P_A.

In the case of conjunctive queries and conjunctive views, this compatibility can be checked in time polynomial in the size of the query expressions. We omit a full proof for lack of space, and only briefly discuss how this could be done for predicates over numerical attributes here: The predicates could be between two attributes or between an attribute and a constant. Group attributes in to equivalence classes based on equality predicates between attributes. Then a pass over the inequality predicates between attributes establishes relationships between these equivalence classes: Create a graph with vertices of the graph corresponding to the equivalence classes. Let E_A be the equivalence class to which attribute A belongs. Then for every constraint of the form $A < B$ or $A \leq B$, create a directed edge from E_A to E_B. If the resulting graph has a directed cycle, the predicates are contradictory since they are conjunctive, and we can stop.

Now the predicates between attributes and constants define sets of intervals of allowed values for each equivalence class. For example the constraints $3 <= A < 7$ and $A \neq 5$ define the intervals $[3, 5), (5, 7)$ for E_A. If the upper and lower bounds defined by two predicates for a given equivalence class have no common intersection, then the predicates are contradictory and we can stop.

Consider a total order of the graph vertices that satisfies the partial order forced by the directed edges (can be found by a topological sort). Start with the smallest vertex in the total order. Assign to it the smallest value it could take, as permitted by its intervals of allowed values and update the intervals of all vertices reachable from it accordingly. If there is no feasible value for this vertex, the predicates must be contradictory and we can stop. Otherwise, we continue to the next vertex in the total order. If at the end, all vertices are assigned values, we know that P_Q and P_A are compatible, and can report suspiciousness of the query batch. One reason this works is because of our assumption on the domain: the value that an attribute can take on is unconstrained by the values of any other attributes.

If we allow $\neq$ constraints between attributes, a hardness result follows due to a reduction from graph coloring:

Theorem 7: Determining exactly if a batch of conjunctive SPJ queries is weakly syntactically suspicious with respect to a conjunctive SPJ audit expression is NP-hard in the size of the query expressions if $\neq$ predicates are allowed between attributes in the `WHERE` clause predicates.

Set vs. Multi-set semantics: Note that none of our definitions of suspiciousness actually required queries or audit expressions to follow set semantics. However the success of our Auditors 1 and 2 in determining the semantic or weak syntactic suspiciousness of a query batch *exactly* required a multi-set semantics. In general, we could use these auditors as such for query batches with a set semantics as well by treating them as multi-set queries, however they would be conservative, i.e., the auditors would some times label query batches as semantically or weak syntactically suspicious even if they would not be under set semantics. The reverse would not be true. Auditor 2 operating in this conservative fashion on conjunctive SPJ queries following set semantics would still provide the association breach detection guarantee of Theorem 8 introduced in the next Section.

We also examined whether Auditor 2 could be used to detect perfect privacy suspiciousness as well in the case of multi-set semantics. But it turns out that the notion of criticality does not allow for such a simple auditor even in this case. This is because a duplicate-preserving query and a view can share a critical tuple even though their `WHERE` clause predicates are not compatible. For example, consider the following query and audit expression over two tables A and B that contain just one column each:

Q:
```
SELECT A.value FROM A, B
WHERE A.value < B.value
```

V:
```
AUDIT A.value FROM A, B
WHERE A.value > B.value
```

Consider an instance $A = \{2, 4\}$ and $B = \{1, 3\}$. The presence or absence of tuple $2 \in A$ makes a difference to the results of both Q and V. 2 is thus critical to both Q and V although their `WHERE` clause predicates are not compatible. Thus Q is suspicious with respect to V under perfect privacy suspiciousness. However no tuple in $A \times B$ in any instantiation of A and B could be indispensable to both Q and V due to incompatibility of their `WHERE` clause predicates. Thus Q is not weakly syntactically suspicious with respect to V. Indeed, one might claim there isn't a compelling reason to deem Q suspicious with respect to V since the user's intent was to find all tuples in A that were smaller than some tuple in B. The only time this is a problem is if he already knew the value of some tuple in A that he didn't see in his query results. But in this case since it is the value of the tuple itself that we seek to hide and the user already knew this value, perhaps it is okay to not consider this a privacy breach. As an aside, [7] develops

algorithms for determining perfect privacy suspiciousness for conjunctive queries that follow set semantics. However they do not work for queries with inequalities such as query Q above.

V. Detecting Association Privacy Breaches

Thus far, we have a definition of suspiciousness for which a large class of conjunctive queries can be audited in time polynomial in the size of the query expressions. An important question that remains is, what kinds of disclosure of the forbidden view are detected by such a weak syntactic auditor. We are unable to give guarantees for arbitrary forbidden views: see the technical report [10] for more of a discussion on this and formal definitions of the notions of full and partial disclosure discussed earlier. However for a specific kind of forbidden view, the weak syntactic auditor gives concrete guarantees.

Specifically, we consider a common kind of disclosure that a database system would like to detect: the disclosure of the association between an individual and his sensitive attributes. Consider a database with m tables. Let $\mathcal{NS}_i$ and $\mathcal{S}_i$ be the so called "non-sensitive" and "sensitive attributes" of the ith table, T_i. Suppose each individual in the database has exactly one tuple in each of these tables. And one such individual, I, has complained that his sensitive attributes were leaked and is therefore being subject to a disclosure review. Then ideally we would like to label a query batch posed in the past as suspicious if it violates the *association privacy* of the individual as defined below.

Definition 9 (Association Privacy Breach): A set of queries $\mathcal{Q}$ violates the association privacy of an individual I, if there exists some attacker with prior distribution D over I's sensitive attributes for whom,

$$Pr_D\{\mathcal{S}(I) = s|\mathcal{Q}\} \neq Pr_D\{\mathcal{S}(I) = s\}$$

Here $\mathcal{S}(\mathcal{I})$ is the concatenation of all the sensitive attributes of individual I.

Note that the distribution D here is the prior distribution of the attacker and the definition of the breach considers all such attackers. On the other hand in the case of perfect privacy, D was the distribution that the data was drawn from and an attacker with a prior very different from the true data distribution could potentially learn a lot from the released views. In this sense, the notion of association privacy of an individual is stronger than the notion of perfect privacy.

If we make certain assumptions about the attacker's prior distribution D and frame an appropriate set of audit expressions, then a weak syntactic auditor can be used to detect association privacy breaches. We frame the following set of audit expressions that we call *association views* for k individuals being subject to a disclosure review:

$\forall\ i\ in\ 1 \ldots m$
AUDIT $\mathcal{S}_i$ FROM T_i
WHERE $\mathcal{NS}_i = ns_i^1 \vee \mathcal{NS}_i = ns_i^2 \vee \ldots \vee \mathcal{NS}_i = ns_i^k$

Here ns_i^j are the values of the non-sensitive attributes for the jth individual in the ith table. Each $\mathcal{NS}_i = ns_i^j$ term is itself a *conjunction* of equality predicates on the non-sensitive attributes of the jth individual.

We make the following (reasonable) assumptions about an attacker's prior D: (1) The attacker knew beforehand that the individuals being subject to the disclosure review were in the database (2) The attacker knew all the non-sensitive attributes of the individuals being subject to the disclosure review (even if he didn't know them beforehand, he should have been allowed to learn them by querying the database) (3) He may or may not have known some of the sensitive attributes of these individuals beforehand as well (4) He had some prior belief about correlations between attributes of each individual, but had no beliefs about correlations between attributes of different individuals.

Now consider a weak syntactic auditor that labels a query batch as suspicious if it is suspicious with respect to any one of the association views [1]. Then:

Theorem 8: If a batch of queries $\mathcal{Q}$ violates the association privacy of some $I \in \mathcal{I}$, a weak syntactic auditor will label $\mathcal{Q}$ as suspicious with respect to the set of association views for the individuals $\mathcal{I}$. Further, only the weakly syntactically suspicious queries within $\mathcal{Q}$ can be responsible for the violation.

We call this the *association breach detection guarantee*. As an example, consider a hospital database containing two tables, one with patients' age, gender, zipcode and disease, the other with patient names and medications. Medication and disease are the sensitive attributes of any individual. Suppose that Alice who is 23 years old and lives in 94305 is in the database and suspects that her medication or disease information was leaked by a query batch. Then the set of association views for Alice are:

```
AUDIT p.disease FROM Patients p
WHERE p.age = 23 ∧ p.gender = 'f' ∧ p.zipcode
= 94305

AUDIT c.medication FROM Customers c
WHERE c.name = 'Alice'
```

A semantic auditor would not be able to give an association breach detection guarantee with respect to the above set of audit expressions. Consider an attacker who knew Alice was in the database. Suppose that based on his prior knowledge of her sensitive attributes and complete knowledge of her non-sensitive attributes, he had known that she took either tylenol or aspirin or advil with equal probability. If she took tylenol, then he knew that she has headaches, if she took advil then he knew that she either has headaches or backaches with equal probability and if she took aspirin then she either has headaches or heart disease with equal probability. Consider

[1] Although each audit expression in the set of association views is not conjunctive, Auditor 2 could easily be adapted to audit conjunctive queries with respect to each disjunctive audit expression in time polynomial in the size of the query expressions by checking for compatibility of the WHERE clause predicates of a query with each disjunct of the WHERE clause of the audit expression.

the following queries:

```
SELECT c.name FROM Customers c
WHERE c.medication = 'tylenol'
```

```
SELECT c.name FROM Customers c
WHERE c.medication = 'advil'
```

If Alice takes aspirin, she is not an indispensable tuple for either query and the query batch would be considered semantically unsuspicious whereas in reality by virtue of multiple negative disclosures, it revealed that Alice did in fact take aspirin and that she either has headaches or heart disease. However a weak syntactic auditor would label the query batch as suspicious. In fact, each query in the batch would be considered suspicious in its own right due to the indispensability of Alice's tuple in some database instance; observe that both queries should in fact be considered suspicious since the answer to either query changes the attacker's distribution on Alice's medication and diseases.

For further discussion on why the association breach detection guarantee does not hold for attackers with prior knowledge of correlations across individuals, and a discussion on other kinds of association views that can be framed, see our technical report at [10].

That said, weak syntactic suspiciousness does not provide as strong disclosure detection guarantees as perfect privacy suspiciousness even in the absence of correlations across individuals. In the example from Section II, the query $\pi_{name}(Patients)$ would be suspicious under perfect privacy in case the secret view is $\pi_{phone}(Patients)$, whereas a weak syntactic auditor would deem it unsuspicious, thereby letting slip the revelation of the size of the forbidden view. But besides forbidden view sizes, is there any other information disclosure that such an auditor would not detect? It would be interesting to characterize exactly the kinds of disclosure detected under the different definitions and to explore other definitions in this space.

VI. Online Auditing

Other than detecting disclosures that have already occurred, a potentially important use of an auditor could be to prevent disclosures from taking place. An auditor could be used to determine whether a query should be *denied* in order to prevent a user from learning information about the forbidden view. Could the auditors we described above be used in an online setting?

The notion of semantic suspiciousness and a semantic auditor don't really make sense in an online setting because the act of denying a query itself could reveal information to a user that the auditor was seeking to protect. For example, consider the following forbidden view:

```
AUDIT p.name, p.disease FROM Patients p
WHERE p.zipcode = 94305
```

Now suppose the following query is permitted by a semantic auditor and it returns only one disease.

```
SELECT p.disease FROM Patients p
WHERE p.zipcode = 94305
```

But the following query is denied

```
SELECT p.name From Patients p
WHERE p.zipcode = 94305 and p.name = 'Alex'
```

The only reason that this could happen is that Alex who lives in 94305 is actually in the database and his disease was the one returned by first query. Even though the second query was denied, the name-disease association of an individual in the forbidden view was revealed. This is because the auditor's reasoning process for the denial involved the answer to the second query which was information that was unavailable to the adversary prior to the denial.

On the other hand, if all we wish to protect is the association privacy of individuals and their sensitive attributes, and are satisfied with the association breach detection guarantee of Theorem 8, a weak syntactic auditor could be used in conjunction with the association views of Section V in an online fashion. Suspicious queries would be denied, unsuspicious queries would be answered, and an attacker would not learn a posterior distribution over the sensitive attributes of any one individual that is different from his prior. In particular, let $\mathcal{Q} = \{q_1, \ldots, q_t\}$ be a sequence of queries and $\mathcal{A} = \{a_1, \ldots, a_t\}$ be the answers supplied by the weak syntactic auditor. Here each a_i is either the true query result of q_i over the database or a denial. Then for each individual $I \in \mathcal{I}$ we can show that for any attacker with a prior distribution D satisfying the conditions enumerated in Section V,

$$Pr_D\{\mathcal{S}(I) = s | q_1, \ldots, q_t, a_1, \ldots, a_t\} = Pr_D\{\mathcal{S}(I) = s\}$$

In this scenario, denials do not leak additional information because the decision to deny a query is not based on the actual database state. Suspicious queries are determined by considering all possible database instances and not by using any information that is unavailable to an adversary. The adversary can himself carry out the auditor's reasoning process and predict when his queries will be denied.

Also note that due to the decomposability result of Theorem 3, suspiciousness of a query can be determined independently of all past queries and thus the auditor need not maintain an audit trail of all past queries posed. In the interests of avoiding a very draconian denial policy, it would be worth exploring relaxations of weak syntactic suspiciousness that permit a bounded change in an attacker's posterior distribution on the sensitive attributes of an individual for an association breach detecting set of audit expressions. This is a very interesting avenue for future research.

VII. Auditing and Access Control

As we have seen, previous work in [8], [2] can be cast in the auditing framework proposed in this paper. The differences lie in the definitions of suspiciousness

In database access control literature, the problem studied is essentially the dual of the auditing problem: Given a set of "authorization views", $\mathcal{V} = \{V_1, \ldots, V_k\}$ for a user and

a definition of *validity*, what queries posed by the user are valid with respect to the authorization views and are therefore safe to answer? Here the authorization views correspond to information that the user is allowed to access and they can be specified via SQL queries similar to audit expressions. One notion of validity considered in particular in [9], [14], [15], [13] is that of unconditional validity:

Definition 10: (Unconditionally Valid Query) Given a set of authorization views, $\mathcal{V}$, a query Q is unconditionally valid with respect to $\mathcal{V}$ if there is a query Q' that can be written using only the instantiated authorization views and is equivalent to Q, i.e., Q' produces the same result as Q on all database instances.

We now formalize the relationship between unconditional validity and suspiciousness of a query. We use the notation $P(t) = 1$ to denote that tuple t satisfies predicates P. We first start off with the following simple result relating unconditional validity to perfect privacy suspiciousness in case of a database containing a single table.

Theorem 9: Given a forbidden view with no self joins $A = \pi_{C_A}(\sigma_{P_A}(T))$, there exists an authorization view over the database such that a query with no self joins $Q = \pi_{C_Q}(\sigma_{P_Q}(T))$ is suspicious with respect to A under perfect privacy suspiciousness iff it is not unconditionally valid with respect to the authorization view.

The intuition behind this and the next result is that the forbidden view implicity defines a set of authorization views that are essentially its complement. In this case the implicit authorization view is $V = \pi_*(\sigma_{\bar{P_A}}(T))$.

We can also show the following result relating unconditional validity to weak syntactic suspiciousness.

Theorem 10: Consider a conjunctive forbidden view $A = \pi_{C_A}(\sigma_{P_A}(\mathcal{T} \times \mathcal{S}))$ with no self joins such that $C_A \cap C_{P_A} = \emptyset$. Here C_{P_A} refers to the columns accessed by P_A.

Consider any conjunctive query $A = \pi_{C_Q}(\sigma_{P_Q}(\mathcal{T} \times \mathcal{R}))$ with no self joins such that if $C_A \cap C_{P_Q} \neq \emptyset$ then if $\exists t \in dom(\mathcal{T} \times \mathcal{R} \times \mathcal{S})$ such that $P_A(t) = 1 \wedge P_Q(t) = 1$ then $\exists t' \in dom(\mathcal{T} \times \mathcal{R} \times \mathcal{S})$ such that $P_A(t') = 1 \wedge P_Q(t') \neq 1$.

There exist a set of authorization views, $\mathcal{V}$, over the database such that every such Q is weakly syntactically suspicious with respect to A iff it is not unconditionally valid w.r.t. $\mathcal{V}$.

The condition on the query in the statement of the theorem essentially requires that the predicates of a query that accesses any of the `audit list` columns of the audit expression should not be vacuous with respect to forbidden tuples, i.e., it shouldn't be the case that all forbidden tuples in the domain of possible tuples satisfy the query predicates. We illustrate this condition and the theorem with an example. Consider the following forbidden view

```
AUDIT p.disease FROM Patients p
WHERE p.zipcode ≠ 94305
```

The forbidden view thus says that a user does not have a right to learn the disease information of any individual living outside of zipcode 94305. The corresponding authorization views would then be

$V1$: `SELECT * FROM Patients p WHERE p.zipcode = 94305`

$V2$: `SELECT * \ p.disease FROM Patients p WHERE p.zipcode ≠ 94305`

Here the $\setminus$ in the `SELECT` clause of $V2$ is the set difference operator, i.e., $V2$ selects all but the disease column. Consider the query:

Q: `SELECT p.zipcode FROM Patients p WHERE p.disease = 'cancer'`

Q is clearly weakly syntactically suspicious with respect to the forbidden view because there could be a database instance where an individual with cancer lives outside of area code 94305. Further Q cannot be rewritten as Q' over $V1$ and $V2$ such that the answer to Q' is the same as the answer to Q for every possible database instance. This is because patients living outside of 94305 could either have or not have cancer (due to our assumptions on the domain) and $V2$ does not provide access to their disease column. Thus Q is not unconditionally valid with respect to $V1$ and $V2$. Now the condition on the query and domain must make some sense. If the domain was known to be such that every single person living outside of 94305 could only have cancer, then the authorization views, the way we have designed them would essentially be hiding no information from a user at all. On the other hand a query such as

```
SELECT p.phonenumber FROM Patients p
```

is not suspicious with respect to the forbidden view and also can be rewritten over the authorization views as

```
SELECT p.phonenumber FROM V1
UNION
SELECT p.phonenumber FROM V2
```

The above theorems are interesting as they tie together work in auditing and work in the database access control literature. Mechanisms that are used for detecting validity of an SPJ query could potentially also be used for detecting suspiciousness of SPJ queries in some cases. Ideally a database management system would try to incorporate both mechanisms - a database access control mechanism that gives users access to various parts of the data, thereby providing utility, and an auditing mechanism (perhaps an online auditing mechanism) to detect or prevent privacy breaches. An interesting avenue for future work would be to see how both these mechanisms could be combined in a system to work together. Checking for consistency in such a system would then be an interesting question, i.e., for a set of forbidden views, a notion of suspiciousness, a set of authorization views and a notion of validity, is it the case that every suspicious query is invalid and every valid query is not suspicious. Another interesting question to ask would be for a given level of privacy (forbidden views and definition of suspiciousness), what is the maximum utility one could have while maintaining consistency and vice versa.

We have only just scratched the surface of this connection so far, and for most definitions and query classes, the questions remains wide open.

VIII. Conclusions

In this paper we introduced a framework for auditing queries and several different notions of suspiciousness that differed in their disclosure detection guarantees as well as the tractability of auditing under them for different classes of queries. Out of all these the notion of weak syntactic suspiciousness had many positive qualities:

- A weak syntactic auditor was designed for a large class of conjunctive queries that was polynomial in just the size of the query expressions.
- The weak syntactic auditor in conjunction with a set of association views can be used to detect queries that potentially violated the association privacy of an individual and his sensitive attributes.
- The weak syntactic auditor can be used in an online setting to deny queries and prevent violations of the association privacy of an individual.

It would be interesting to explore other definitions of suspiciousness that lie in this space and their properties. In particular, a clear future direction would be to consider relaxations of weak syntactic suspiciousness that in conjunction with a set of association views permit bounded changes in an attacker's posterior distribution over an individual's sensitive attributes. This would be very useful in an online setting to avoid draconian denial policies. We also drew an interesting connection to the problem of checking query validity. Exploring this connection further is an avenue for further research.

Acknowledgements

We thank Dan Suciu, Gerome Miklau and Nilesh Dalvi for their help in understanding results from [8]. This work was supported in part by NSF Grant ITR-0331640, TRUST (NSF award number CCF-0424422), and a grant from Google. The second author was also supported in part by a Stanford Graduate Fellowship from Sequoia Capital.

References

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

[2] R. Agrawal, R. Bayardo, C. Faloutsos, J. Kieman, R. Rantzau, and R. Srikant. Auditing Compliance with a Hippocratic Database. In *Proceedings of the International Conference on Very Large Databases (VLDB)*, 2004.

[3] F. Chin. Security Problems on Inference Control for SUM, MAX, and MIN Queries. *J. ACM*, 33(3):451–464, 1986.

[4] F. Y. Chin, P. Kossowski, and S. C. Loh. Efficient Inference Control for Range Sum Queries. *Theoretical Computer Science*, 32(1–2):77–86, July 1984.

[5] D. Dobkin, A. Jones, and R. Lipton. Secure Databases: Protection against User Influence. *ACM Transactions on Database Systems (TODS)*, 4(1):97–106, 1979.

[6] K. Kenthapadi, N. Mishra, and K. Nissim. Simulatable Auditing. In *Proceedings of the ACM Symposium on Principles of Database Systems (PODS)*, pages 118–127, 2005.

[7] A. Machanavajjhala and J. Gehrke. On the Efficiency of Checking Perfect Privacy. In *Proceedings of the ACM Symposium on Principles of Database Systems (PODS)*, 2006.

[8] G. Miklau and D. Suciu. A Formal Analysis of Information Disclosure in Data Exchange. *Journal of Computer and System Sciences*, 2006.

[9] A. Motro. An Access Authorization Model for Relational Databases Based on Algebraic Manipulation of View Definitions. In *Proceedings of the International Conference on Data Engineering (ICDE)*, 1989.

[10] R. Motwani, S. U. Nabar, and D. Thomas. Auditing SQL Queries. *http://dbpubs.stanford.edu/pub/2007-13, Technical Report, Stanford University*, 2007.

[11] S. U. Nabar, B. Marthi, K. Kenthapadi, N. Mishra, and R. Motwani. Towards Robustness in Query Auditing. In *Proceedings of the International Conference on Very Large Databases (VLDB)*, 2006.

[12] S. Reiss. Security in Databases: A Combinatorial Study. *J. ACM*, 26(1):45–57, 1979.

[13] S. Rizvi, A. Medelzon, S. Sudarshan, and P. Roy. Extending Query Rewriting Techniques for Fine-Grained Access Control. In *Proceedings of the ACM International Conference on Management of Data (SIGMOD)*, 2004.

[14] A. Rosenthal and E. Sciore. View Security as the Basis for Data Warehouse Security. In *Proceedings of the International Workshop on Design and Management of Data Warehouses*, 2000.

[15] A. Rosenthal, E. Sciore, and V. Doshi. Security Administration for Federations, Warehouses, and other Derived Data. In *Proceedings of the IFIP WG11.3 Conference on Database Security*, 1999.

[16] L. Sweeney. Uniqueness of simple demographics in the U.S. population. *LIDAP-WP4. Carnegie Mellon University, Laboratory for International Data Privacy, Pittsburgh, PA*, 2000.