NAIST-IS-DD1261020

Doctoral Dissertation

Verification Methods for Security against Inference Attacks on XML and Relational Databases

Chittaphone Phonharath

June 18, 2015

Department of Information Systems
Graduate School of Information Science
Nara Institute of Science and Technology

A Doctoral Dissertation
submitted to Graduate School of Information Science,
Nara Institute of Science and Technology
in partial fulfillment of the requirements for the degree of
Doctor of ENGINEERING

Chittaphone Phonharath

Thesis Committee:

Professor Minoru Ito	(Supervisor)
Professor Kenichi Matsumoto	(Co-supervisor)
Professor Hiroyuki Seki	(Co-supervisor)
Associate Professor Yuichi Kaji	(Co-supervisor)
Assistant Professor Kenji Hashimoto	(Nagoya University)

Verification Methods for Security against Inference Attacks on XML and Relational Databases*

Chittaphone Phonharath

Abstract

Access control is one of the most important mechanisms of a database management system to keep confidential information secret, which should be protected from malicious users (or attackers). A typical way of realizing access control is to divide the available set of queries into authorized queries and unauthorized ones, and to allow a user to obtain only the result of authorized queries for a database instance. At first glance, this access control policy works well. However, it may happen that a user can obtain the result (or a set of candidates of the correct result) of an unauthorized query by cleverly combining the results of authorized queries. Such an attack is called an *inference attack*. This dissertation presents two studies of verifying the security against inference attacks on both *XML* and *relational* databases.

Firstly, we study a static analysis problem on k-secrecy, which is a metric for the security against inference attacks on **XML databases**. Intuitively, k-secrecy means that the number of candidates of sensitive data of a given database instance or the result of an unauthorized query cannot be narrowed down to $k-1$ by using available information such as authorized queries and their results. We investigate the decidability of the *schema-level k-secrecy* problem defined as follows: for a given XML database schema, an authorized query and an unauthorized query decide whether every database instance conforming to the given schema is k-secret. We first show that the schema-level k-secrecy problem is undecidable for

*Doctoral Dissertation, Department of Information Systems, Graduate School of Information Science, Nara Institute of Science and Technology, NAIST-IS-DD1261020, June 18, 2015.

any finite $k > 1$ even when queries are represented by a simple subclass of linear deterministic top-down tree transducers (LDTT). Next, we give an algorithm for deciding the schema-level ∞-secrecy problem and analyze its time complexity. We show the schema-level ∞-secrecy problem is EXPTIME-complete for LDTT. Moreover, we show similar results for LDTT with regular look-ahead.

Secondly, we propose a notion of the security against inference attacks by extending ℓ-diversity in **relational databases** with access control for queries. More specifically, we propose a new privacy notion called *query-based ℓ-diversity*. A database instance T is (query-based) ℓ-diverse with respect to given authorized queries if an attacker cannot narrow down the number of possible values of the sensitive information to less than ℓ by inference using the result of the authorized queries on the instance T and the meaning of the queries. We provide two approaches to deciding the property. The first approach directly decides the problem for a given input by using a relational database management system, e.g. Structured Query Language (SQL). The second approach transforms a given input to a logical formula and decides the problem by model counting using a #SAT solver. We discuss the effectiveness and scalability of two approaches based on the experimental results.

Keywords:

Database Security, Privacy, Diversity, Inference Attacks, Relational Databases, XML Databases.

Contents

1 Introduction **1**

1.1. Research Motivation 1

1.2. Research Contribution 4

1.2.1 Contribution on XML Databases 4

1.2.2 Contribution on Relational Databases 5

1.3. Dissertation Layout 6

2 Related Work **7**

2.1. k-Anonymity 7

2.2. ℓ-Diversity 8

2.3. k-Secrecy 10

3 Deciding Schema k-Secrecy on XML Databases **12**

3.1. Introduction 12

3.2. Models 13

3.2.1 XML Documents 13

3.2.2 XML Databases 14

3.2.3 Queries 15

3.3. Schema k-Secrecy Problem 17

3.4. Undecidability Result 19

3.5. Decidability Results on ∞-SS 21

3.5.1 Overview of Decision Algorithm 21

3.5.2 Detailed Construction 23

3.5.3 Correctness 28

3.5.4 Time Complexity 31

3.5.5 Extension to LDTT^R . . . 34
3.6. Discussion . . . 35
3.6.1 Multiple Authorized Queries . . . 35
3.6.2 k-Secrecy Preservation through Updates . . . 36
3.7. Conclusion . . . 37

4 Query-Based ℓ-Diversity on Relational Databases 38
4.1. Introduction . . . 38
4.2. Models . . . 39
4.3. Proposed Framework . . . 40
4.4. Verification by Relational Algebra . . . 42
4.5. Verification by Model Counting . . . 44
Constructing Constraint . . . 45
Counting Candidates . . . 47
4.6. Experiments . . . 47
4.6.1 Experimental Result of Relational Algebra . . . 47
Setup . . . 47
Datasets and Queries . . . 48
4.6.2 Experimental Result of Model Counting . . . 49
Setup . . . 49
Datasets and Queries . . . 49
4.6.3 Additional Result . . . 50
Setup . . . 50
Datasets and Queries . . . 50
4.7. Conclusion . . . 52

5 Conclusion 53

Acknowledgements 56

References 58
References . . . 58

Publication List 61

List of Figures

1.1 An inference attack. 4

2.1 An attack on XML instance. 11

3.1 Trees accepted by A_G. 19
3.2 Reduction from PCP to 2-SS. 21
3.3 Deleted and undeleted nodes by T. 22
3.4 The dependency graph G_{A_I}. 27
3.5 Lemma 3.5.2. 28
3.6 Lemma 3.5.3. 29
3.7 Lemma 3.5.4. 30

List of Tables

2.1 A 2-anonymous instance. 8
2.2 A sample instance. 9
2.3 A 3-anonymous and 2-diverse instance. 9

3.1 Schema ∞-secrecy for Example 3.3.3. 19
3.2 The execution results of the algorithm for Example 3.3.3. 26

4.1 Total time of verifying 2-diversity. 48
4.2 Performance of model counting method. 50
4.3 Scalability of model counting method. 50
4.4 Total running time of relational algebra. 51
4.5 Computation time of two approaches on a dataset. 51

Chapter 1

Introduction

Database security is one of the most important challenges in minimizing the leakage of sensitive information due to accesses or to data publishing, which have been growing rapidly and more powerful. Databases may store large amount of sensitive information of individuals or organizations. The problem occurring is that data publishing is vulnerable to attacks, which can lead to serious problems such as privacy information leakage. Database security is built upon a framework of three structures: confidentiality, integrity and availability [2]. Confidentiality refers to the protection of data against an unauthorized access. Integrity refers to the prevention of unauthorized modification. Availability refers to the prevention and recovery from hardware and software errors as well as from malicious data access resulting in the denial of data availability.

1.1. Research Motivation

Access control is a means to allow or deny users to do operations such as read, write, modify, etc. on data in a computer system. The system decides whether to allow or deny a user's access requests to information or resources in the system, and if a request is allowed, then that user can do operations according to the access control rules and roles of the user. Access control rules are determined based on information rights, security policy, authorization rules and other factors depending on the system environment. Furthermore, access control is a traditional mechanism for confidentially restricting accesses to a database made by a user by

dividing the queries into authorized and unauthorized ones, and restricting the portion of the data that can be retrieved and updated by the user. However, it may happen that a user can obtain the result (or a set of candidates of correct result) of an unauthorized query by cleverly combining the results of authorized queries. Such an attack is called an *inference attack* as described below.

Even though a database management system (DBMS for short) provides an appropriate access control mechanism, there is a possibility that sensitive information (unauthorized by access control) can be indirectly obtained from available information (authorized by access control). Let us take a relational database instance r with a schema (student_name, year, salary, TA_subject) as an example and assume that two queries $q_1(r) = \pi$(student_name, TA_subject)(r) and $q_2(r) = \pi$(salary, TA_subject)(r) are authorized and $q_3(r) = \pi$(student_name, salary, TA_subject)(r) is not authorized (because the income of an individual is private information) where $\pi(X)(r)$ means the projection of r onto X. However, if we take the natural join of $q_1(r)$ and $q_2(r)$, then we can obtain a superset of $q_3(r)$, which is a candidate set of the result of the unauthorized query q_3 on r. As shown in this example, an inference attack is a way to infer sensitive information from available information such as the result of authorized queries, the code (meaning) of the authorized and unauthorized queries, and possibly, some external information. If the number of candidates of the result of an unauthorized query is small, then the attacker may obtain information sufficient for his purpose. In the above example, if there are only two candidates (Alice, 100, Database) and (Alice, 70, Database), the attacker knows that the salary of Alice for Database is at least 70.

As shown in the above example, the number of candidates of the sensitive information depends on how secure the system should be. We need a quantitative notion to measure the security or privacy of the system under consideration. There are a few well-known quantitative notions for database privacy, k-anonymity [19][21], ℓ-diversity [15] and t-closeness [14]. These notions assume the following basic concepts on relational databases. The set of attributes are divided into sensitive and nonsensitive attributes. Also, a subset of the nonsensitive attributes, called the quasi-identifier, is assumed. The value of the quasi-identifier is potentially used to identify the tuple of a target individual by linking the dis-

closed information with external data. A database instance satisfies k-anonymity if for any value of the quasi-identifier, there are k or more tuples having that value of the quasi-identifier. However, a k-anonymous database may still have some issues because the database may lack the diversity in the sensitive attributes. ℓ-diversity has been proposed by [15] to overcome the weakness of k-anonymity. Though [15] proposes a general definition of ℓ-diversity, we just review a simple and frequently used one, called distinctive non-recursive ℓ-diversity (or simply, ℓ-diversity). A database instance satisfies ℓ-diversity if for each equivalence class, all tuples within the same class have the same value of the quasi-identifier, there are at least ℓ different values of the sensitive attributes.

Recently, Hashimoto, et al. [11] proposed a quantitative security notion called instance-level k-secrecy to quantify the security against inference attacks on XML databases. Assume that a database instance T of a schema $\mathbf{R}$, authorized queries $q_1, q_2, \ldots, q_m$ and an unauthorized query q_s are given. An attacker knows $\mathbf{R}$, $q_1, q_2, \ldots, q_m, q_s$ (the meaning of the authorized and unauthorized queries) and $q_1(T), q_2(T), \ldots, q_m(T)$ (the result of the authorized queries), but he does not know T. The goal of the attacker is to obtain $q_s(T)$, which is the result of the unauthorized query q_s on T (as shown in Example 1.1). For a positive integer k, a database instance T is k-secret with respect to $\mathbf{R}, q_1, \ldots, q_m, q_s$ if the attacker cannot narrow down the number of the candidates of $q_s(T)$ to less than k. T is ∞-secret if the candidates of $q_s(T)$ are infinite.

> **Example 1.1.1.** Figure 1.1 shows the scenario that an attacker infers the sensitive information $q_s(T)$ (on the lower left) of a database instance T by combining the result of queries $q_1(T), q_2(T), ..., q_m(T)$ that gives the candidate set of sensitive information $q_s(T)$ (on the lower right), which means one of the candidates included in this set is the correct sensitive information which is $q_s(T)$ (on the lower left). Assume that $q_1, q_2, ..., q_m$ are authorized queries and q_s is an unauthorized query.

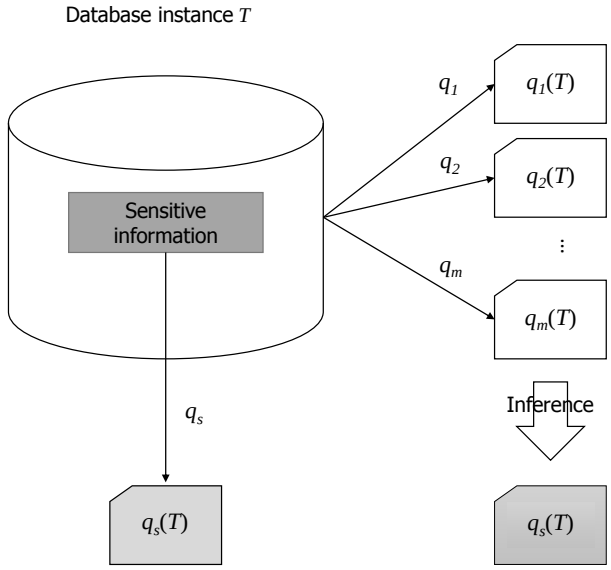

Figure 1.1. An inference attack.

[11] showed that k-secrecy is decidable for XML databases where queries are given as tree transducers in a certain subclass that can use relabeling and deletion.

1.2. Research Contribution

The main purpose of this dissertation is to propose methods for verifying security against inference attacks on XML and relational databases. The following subsections give the summary of the contribution of this thesis.

1.2.1 Contribution on XML Databases

To extend the idea of instance-level k-secrecy [11], we introduce the notion of schema-level k-secrecy. For a positive integer k, a schema $\mathbf{R}$ is k-secret with respect to authorized queries $q_1, ..., q_m$ and an unauthorized query q_s if every instance conforming to $\mathbf{R}$ is k-secret with respect to those queries. If we can

guarantee that every instance is k-secret, then we do not need to repeatedly test whether a new instance is k-secret or not every time an instance is updated. In this study, we restrict the number of authorized queries to one for simplicity. We show that the schema-level k-secrecy is undecidable for any finite k when queries are given by linear deterministic top-down tree transducers. Also, we show that the schema-level ∞-secrecy is decidable for the same class of queries and investigate the computational complexity of the problem. The details of this study will be discussed in Chapter 3.

1.2.2 Contribution on Relational Databases

To apply the idea of instance-level k-secrecy [11] in defining a quantitative notion for anonymity, this study introduces a notion of the security against inference attacks by extending ℓ-diversity [15] in relational databases with access control for queries. More specifically, we propose a new privacy notion called *query-based ℓ-diversity*. A database instance T is (query-based) ℓ-diverse with respect to given authorized queries if an attacker cannot narrow down the number of possible values of the sensitive information for any individual to less than ℓ by inference, based on the result of the authorized queries on the instance T and the queries themselves. Two approaches are proposed and implemented by Structured Query Languages (SQL) and #SAT solver, respectively. Also, we compare the scalability and effectiveness of the two approaches based on the experiments conducted on the implemented tools. The first approach is based on relational algebra and restricts the authorized queries only on projection queries (selection is not allowed). The second approach transforms a given input of the problem to a logical formula and decides the problem by model counting using #SAT solver, where queries are self-join free conjunctive queries, consisting of projection, selection and join without self-join. The details of this study will be presented in Chapter 4.

1.3. Dissertation Layout

This dissertation is organized as follows:

Chapter **1** presents the purpose by introducing the motivations of this dissertation. Mainly, k-secrecy, the verification notion on XML databases is discussed. Also, the research contributions are briefly introduced to both XML and relational databases. Chapter **2** presents the related security and privacy notions in details. Chapter **3** extends the notion of instance-level k-secrecy to schema-level k-secrecy. The decidability and computational complexity of the problem of schema-level k-secrecy are investigated for the class of queries represented by linear deterministic top-down tree transducers. Chapter **4** gives the proposal of the query-based ℓ-diversity, which is a notion obtained by combining k-secrecy and ℓ-diversity in the relational database setting. We propose two approaches to the problem of verifying the query-based ℓ-diversity. One of them based on relational algebra and the other one is based on model counting. Then we end with the experimental results conducted on two verification tools implemented by SQL and #SAT solver. Chapter **5** provides the conclusion of this dissertation.

Chapter 2

Related Work

In the introduction, we briefly discussed about an inference attack, which is the main problem of this dissertation. Furthermore, we also mentioned access control as a traditional mechanism that confidentially restricts accesses to a database. Finally, we discussed about security and privacy notions such as k-anonymity, ℓ-diversity and k-secrecy. This chapter presents these notions in more details.

2.1. k-Anonymity

A database instance satisfies k-anonymity if for any value of the quasi-identifier, there are k or more tuples having that value of the quasi-identifier. A maximal subset of tuples having same values of the quasi-identifier is called an equivalence class. k-anonymity means that the cardinality of each equivalence class is at least k. A transformation of a given instance to another instance satisfying k-anonymity is called a k-anonymization for the original instance. [21] proposed a method for k-anonymization by hiding some information of individuals by generalization and suppression. Generalization replaces a value with a less specific but semantically consistent value, while suppression hides the data or does not release the entire data. Various anonymization methods have been reported using clustering, branch-and-bound search and so on [1][4]. k-anonymity is a simple notion and has been frequently used.

k-anonymity guarantees that an individual's information cannot be distinguished from a given number of people in the record of size k. In general, the

higher the value of k, the more privacy is achieved.

Example 2.1.1. Consider a database instance in Table 2.1, which contains two attributes; Zipcode and Age. The instance is k-anonymous if for any record, $k-1$ other records have the same Zipcode and Age. Table 2.1 satisfies 2-anonymity.

Table 2.1. A 2-anonymous instance.

Zipcode	Age
130-****	2*
130-****	2*
130-****	2*
130-****	2*
630-****	2*
630-****	2*

As discussed in [15], however, a k-anonymous database may still have some issues because the database may lack the diversity in the sensitive attributes. ℓ-diversity has been proposed by [15] to overcome the weakness of k-anonymity.

2.2. ℓ-Diversity

ℓ-diversity has been proposed by [15] to overcome the weakness of k-anonymity and improves anonymization beyond what k-anonymity provides. [24] showed that ℓ-diversity always guarantees stronger privacy preservation than k-anonymity. The difference between these two notions is that while k-anonymity requires each equivalence classes of quasi-identifier to have k entires, ℓ-diversity requires that there are ℓ different sensitive values for each quasi-identifier, as shown in Table 2.3. Though [15] proposes a general definition of ℓ-diversity, we review a simple and frequently used one, called distinctive non-recursive ℓ-diversity (or simply, ℓ-diversity). A database instance satisfies ℓ-diversity if for each equivalence class, there are at least ℓ different values of the sensitive attributes. Distinctive ℓ-diversity shows that sensitive values are distinctive from each other for at least ℓ values in the same equivalence class.

Example 2.2.1. Table 2.2 shows a database instance consisting of six tuples. Assume that {Zipcode, Gender, Age} is the quasi-identifier and Diagnosis is the sensitive attribute. All the tuples have different values of the quasi-identifier and hence this database instance does not satisfy k-anonymity for any $k \geq 2$.

Assume that in this database instance, the latter four digits of the values of "Zipcode" are hidden, the values of "Gender" are hidden and the values of "Age" are generalized to the intervals of ten years. Then we obtain the database instance shown in Table 2.3. This instance consists of two equivalence classes and the number of tuples in each class is three. Hence, this instance satisfies 3-anonymity and the above transformation is a 3-anonymization for the original instance. Also, the first class has three different values of the sensitive attribute and the second class has two different values. Hence, the transformed instance satisfies 2-diversity but does not satisfy ℓ-diversity for any $\ell \geq 3$.

Table 2.2. A sample instance.

Zipcode	Gender	Age	Diagnosis
123-4567	F	45	A
123-5235	F	44	B
123-4567	F	44	C
378-2102	M	65	A
378-2102	M	62	B
378-2102	F	65	A

Table 2.3. A 3-anonymous and 2-diverse instance.

Zipcode	Gender	Age	Diagnosis
123-****	-	[40,49]	A
123-****	-	[40,49]	B
123-****	-	[40,49]	C
378-****	-	[60,69]	A
378-****	-	[60,69]	B
378-****	-	[60,69]	A

2.3. k-Secrecy

A related but different quantitative notion on database security is given in [11] based on access control on queries. Assume that a database instance T of a schema $\mathbf{R}$, authorized queries $q_1, q_2, \ldots, q_m$ and an unauthorized query q_s are given. An attacker knows $\mathbf{R}$, $q_1, q_2, \ldots, q_m, q_s$ (the meaning of the authorized and unauthorized queries) and $q_1(T), \ldots, q_m(T)$ (the result of the authorized queries), but he does not know T. The goal of the attacker is to obtain $q_s(T)$, which is the result of the unauthorized query q_s on T. For a positive integer k, a database instance T is k-secret with respect to $\mathbf{R}, q_1, \ldots, q_m, q_s$ if the attacker cannot narrow down the number of the candidates of $q_s(T)$ to less than k . T is ∞-secret if the candidates of $q_s(T)$ are infinite. [11] showed that k-secrecy is decidable for XML databases where queries are given as tree transducers in a certain subclass that can use relabeling and deletion. Example 2.3.1 shows the example of an inference attack on an XML database instance.

> **Example 2.3.1.** Figure 2.1 illustrates the example of an inference attack on an XML database instance. Assume that an attacker wants to know the salary (sensitive information of an individual) of an employee named Taro, whose lives in Nara, but an attacker does not have permission to access to such information. Meanwhile, there are some available information which has been retrieved by authorized queries such as the results of queries q_1 (a pair of "Name" and "Address"), and q_2 (a pair of "Salary" and "Address"). An attacker obtains the sensitive information by combining these available information and narrowing down candidate set of sensitive information $Y(t)$. An attacker knows that one of the candidates set $Y(t)$ is the correct information of Taro or the salary of Taro is at least \$1000.

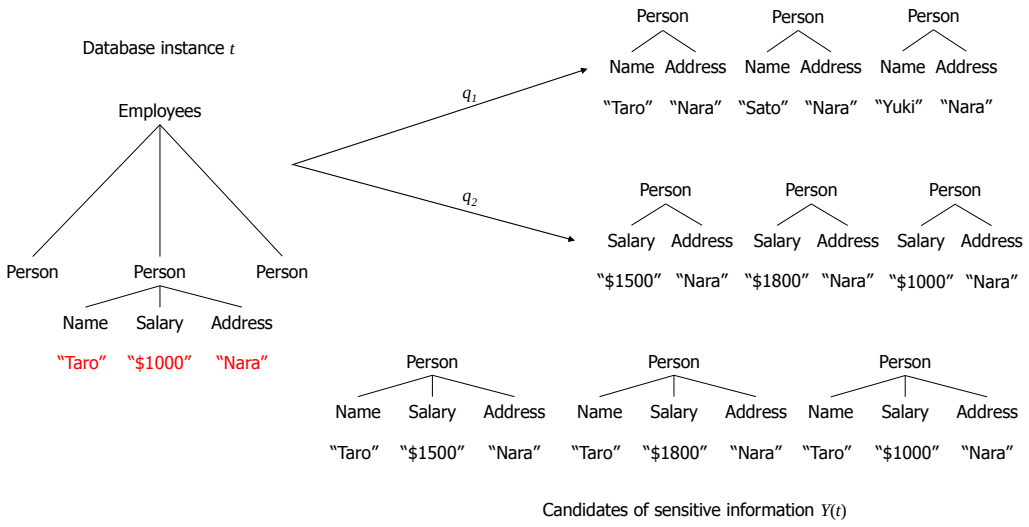

Figure 2.1. An attack on XML instance.

Although [11] deals with XML databases, the notion of k-secrecy is general enough for other kinds of databases. Furthermore, [10] discussed verification of the security against inference attacks on XML databases, considering a functional dependency when unauthorized query is represented by a deterministic top-down tree transducer. More sophisticated notions have been also proposed. For example, [6][16] proposed stronger notions where the probability distribution of possible secrets does not change after observing (authorized) information.

Chapter 3

Deciding Schema k-Secrecy on XML Databases

This chapter presents the notion of schema-level k-secrecy, which is the extension of instance-level k-secrecy. The definition and restriction of the problem are presented. Also, the decidability results and its computational complexity are discussed.

3.1. Introduction

Access control is one of the most important mechanisms of a database management system to keep confidential information secret, which should be protected from malicious access. A typical way of realizing access control is to divide the available set of queries into authorized queries and unauthorized ones and to allow a user to obtain only the result of the authorized queries for a database instance.

At first glance, this access control policy works well. However, it may happen that a user can obtain the result (or a set of candidates of the correct result) of an unauthorized query by cleverly combining the results of authorized queries. Such an attack is called an *inference attack*. Hence, it is desirable to be able to decide whether an inference attack is possible or not for a given database. To quantify the degree of safeness against inference attacks, Hashimoto, et al. [11] define k-secrecy. Intuitively, a database instance is *k-secret* if an attacker cannot narrow down the number of candidates of the result of an unauthorized query to less

than k. In particular, an instance is ∞-secret if the number of candidates cannot be narrowed down to a finite value. The underlying idea is similar to ℓ-diversity in [15]. For a given XML database schema A_G, a database instance t conforming to the schema, an authorized query T_a and an unauthorized query T_s, we say that t is *k-secret* if $|Y(t)| \geq k$ where $Y(t) = \{T_s(t') \mid t' \in L(A_G), T_a(t') = T_a(t)\}$ and $L(A_G)$ is the set of instances conforming to A_G. In particular, t is ∞-secret if $Y(t)$ is infinite. A_G is given by a tree automaton, and queries T_a and T_s are given by tree transducers. In [11], it is shown that k-secrecy is decidable and its time complexity is polynomial in the size of a given instance t if each query is represented by a composition of relabeling and/or deleting tree transducers. The problem discussed in [11] is in instance-level in the sense that we are required to decide whether a *given instance* t is k-secret or not. In this study, we discuss the decidability of the problem for *schema-level k-secrecy* (or simply schema k-secrecy), which is the problem of deciding whether *every instance* conforming to a given schema is k-secret. If we can guarantee that every instance is k-secret, then we do not need to repeatedly test whether a new instance is k-secret or not every time an instance is updated.

3.2. Models

3.2.1 XML Documents

XML documents or XML database instances are often modeled by *unranked labeled ordered trees*, where each node has an arbitrary number of child nodes. Moreover, we do not consider attributes as the input of k-secrecy problem. In this chapter, for simplicity, we consider only *binary* labeled ordered trees. Some encodings from unranked trees to binary trees, without loss of expressive power, are proposed [5][9]. Among them, we adopt *First-Child-Next-Sibling* encoding [9], which is a simple and widely accepted encoding. The first child of a node in an unranked tree corresponds to the left child of the node in the encoded binary tree, and the next sibling of a node in an unranked tree corresponds to the right child of the node in the encoded binary tree. We use # as the special symbol which means that there is no child or no next sibling.

The set $\mathcal{T}_\Sigma$ of (binary) labeled ordered trees over an alphabet Σ is the smallest set defined by:

- $\# \in \mathcal{T}_\Sigma$, and
- $a(t_1, t_2) \in \mathcal{T}_\Sigma$ if $t_1, t_2 \in \mathcal{T}_\Sigma$ and $a \in \Sigma$.

For a tree $t \in \mathcal{T}_\Sigma$, let $V(t)$ denote the set of nodes of t, and let $\lambda(v)$ denote the label of a node $v \in V(t)$. For $t, t' \in \mathcal{T}_\Sigma$ and $v \in V(t)$, let t/v denote the subtree of t rooted at v and let $t[v \leftarrow t']$ denote the tree obtained from t by replacing t/v by t'. The size of a tree t is the number of the nodes of t. Let $\mathcal{X}_2$ be the set consisting of two variables x_1 and x_2. A *context* C over Σ is a tree over $\Sigma \cup \mathcal{X}_2$ such that x_1 and x_2 can only appear on leaves of the tree at most once, respectively. Let $\mathcal{C}(\Sigma, \mathcal{X}_2)$ be the set of all contexts over Σ. For a context $C \in \mathcal{C}(\Sigma, \mathcal{X}_2)$ and $t_1, t_2 \in \mathcal{T}_\Sigma$, let $C[t_1, t_2]$ denote the tree in $\mathcal{T}_\Sigma$ obtained from C by replacing each variable x_i by t_i for $i \in \{1, 2\}$.

3.2.2 XML Databases

A database schema describes syntactic restriction on database instances. Schema for XML documents are modeled by tree automaton.

Definition 3.2.1. A tree automaton (TA) A is a 4-tuple (Q, Σ, Q_0, δ), where Q is a finite set of states, Σ is an alphabet, $Q_0 \subseteq Q$ is the set of initial states, and δ is a set of transition rules in the form of either $q \to a(q_1, q_2)$ or $q \to \#$, where $q, q_1, q_2 \in Q$ and $a \in \Sigma$.

A tree $t \in \mathcal{T}_\Sigma$ is *accepted* by A if and only if there is a mapping r from $V(t)$ to Q such that

- $r(v_0) \in Q_0$ where v_0 is the root of t
- $r(v) \to \lambda(v)(r(v_1), r(v_2)) \in \delta$ if v_1 and v_2 are the left and right children of v, respectively and
- $r(v) \to \# \in \delta$ if v is a leaf.

The mapping r is called a *successful run* of A on t. A successful run will be called a *run* in this study. We define the language $L(A)$ to be $\{\, t \in \mathcal{T}_\Sigma \mid A$ accepts $t \,\}$. $L(A)$ is called the language recognized by A. A *regular tree language* is a language recognized by a TA. The size of a TA A is the number of the states and transition rules of A. A TA A is *unambiguous* if for every $t \in L(A)$, a run of A on t is unique.

For a TA $A = (Q, \Sigma, Q_0, \delta)$, define the dependency graph of A as the directed graph $G_A = (Q, E_A)$ where $E_A = \{(q, q') \mid \exists q'' \in Q, \exists\ a \in \Sigma, q \to a(q', q'') \in \delta$ or $q \to a(q'', q') \in \delta\}$. A path of a directed graph is *nonempty* if it consists of at least one edge. For a TA $A = (Q, \Sigma, Q_0, \delta)$, we say that a state $q \in Q$ is *recursive* through a nonempty path of nodes (states) $q_1, q_2, \ldots, q_n$ $(n \geq 2)$ of G_A such that $q_1 = q_n = q$.

3.2.3 Queries

As languages for XML queries/transformations, XQuery and XSLT are developed and recommended by W3C (World Wide Web Consortium). Both queries and programs written in XQuery and XSLT can be regarded as transformations from trees to trees. Unfortunately, many static analysis problems are undecidable for the full class because XQuery and XSLT are Turing complete [12]. The core of tree transduction of XSLT can be modeled by *tree transducers* [3][5][9]. Tree transducers are a finite machine model for tree transformations. As query models, we use linear deterministic top-down tree transducers (LDTT) and linear deterministic top-down tree transducers with regular look-ahead (LDTT^R).

Definition 3.2.2. A *linear top-down tree transducer* (LTT) T is a quintuple $(Q, \Sigma, \Delta, Q_0, \delta)$, where Q is a finite set of states, Σ is an input alphabet, Δ is an output alphabet, $Q_0 \subseteq Q$ is the set of initial states, and δ is a set of transduction rules of the following two types:

$$q(a(x_1, x_2)) \to C[q_1(x_1), q_2(x_2)] \quad \text{or} \quad q(\#) \to t'$$

where $a \in \Sigma$, $q, q_1, q_2 \in Q$, $C \in \mathcal{C}(\Delta, \mathcal{X}_2)$, and $t' \in \mathcal{T}_\Delta$.

The move relation $\to$ of T is defined as follows:

- If $q(a(x_1, x_2)) \to C[q_1(x_1), q_2(x_2)] \in \delta$, $t_1, t_2 \in \mathcal{T}_\Sigma$ and $t/v = q(a(t_1, t_2))$ then, $t \to t[v \leftarrow C[q_1(t_1), q_2(t_2)]]$ and

- If $q(\#) \to t' \in \delta$ and $t/v = q(\#)$ then $t \to t[v \leftarrow t']$.

The reflexive and transitive closure of $\to$ is denoted by $\to^*$. The transformation induced by T, also denoted as T, is the relation defined by: $T = \{(t, \tau) \mid q_0(t) \to^* \tau, t \in \mathcal{T}_\Sigma, \tau \in \mathcal{T}_\Delta, q_0 \in Q_0\}$. The domain $Dom(T)$ of T is the set $\{t \in \mathcal{T}_\Sigma \mid \exists \tau \in \mathcal{T}_\Delta.(t, \tau) \in T\}$. The image of a subset $L \subseteq \mathcal{T}_\Sigma$ by T is the set $T(L) = \{\tau \in \mathcal{T}_\Delta \mid \exists t \in L.(t, \tau) \in T\}$. The size of an LTT T is the sum of the number of the states and the sizes of the transduction rules of T. The size of a transduction rule ρ is the size of the context in the right-hand side of ρ. An LTT $T = (Q, \Sigma, \Delta, Q_0, \delta)$ is *deterministic* (denoted LDTT) if (1) Q_0 is a singleton and (2) for each state q and $a \in \Sigma$, there is at most one rule that contains q and a in its left-hand side. When T is deterministic, we can regard T as a function from $Dom(T)$ to $\mathcal{T}_\Delta$. Also we let $T^{-1}(t') = \{t \mid T(t) = t'\}$.

The regular look-ahead [7] allows that, when processing each node in top-down manner, it checks whether the subtrees rooted by its children are contained in specified regular tree languages and then processes the node if the containment holds. This ability of look-ahead corresponds to the XPath filters used in XSLT programs.

Definition 3.2.3. A *linear top-down tree transducer with regular look-ahead* (LTT^R) T is a quintuple $(Q, \Sigma, \Delta, Q_0, \delta)$, where Q is a finite set of states, Σ is an input alphabet, Δ is an output alphabet, $Q_0 \subseteq Q$ is the set of initial states, and δ is a set of transduction rules of the following two types:

$$q(a(x_1 : A_1, x_2 : A_2)) \to C[q_1(x_1), q_2(x_2)] \quad \text{or}$$

$$q(\#) \to t'$$

where $a \in \Sigma$, $q_1, q_2, q \in Q$, $C \in \mathcal{C}(\Delta, \mathcal{X}_2)$, $t' \in \mathcal{T}_\Delta$, and A_1, A_2 are TAs over Σ.

The move relation $\to$ of T is defined as follows:

- $t \to t[v \leftarrow C[q_1(t_1), q_2(t_2)]]$ if $q(a(x_1 : A_1, x_2 : A_2)) \to C[q_1(x_1), q_2(x_2)] \in \delta$, $t_1 \in L(A_1)$, $t_2 \in L(A_2)$ and $t/v = q(a(t_1, t_2))$ and

- $t \to t[v \leftarrow t']$ if $q(\#) \to t' \in \delta$ and $t/v = q(\#)$.

The transformation induced by T is defined in the same way as LTT. The size of an LTT^R T is the sum of the number of the states and the sizes of the transduction rules of T. The size of a transduction rule ρ is the sum of the size of the context in the right-hand side and the sizes of the TAs in the left-hand side. An LTT^R T is *deterministic* (denoted LDTT^R) if (1) Q_0 is a singleton and (2) there are no two different rules $q(a(x_1 : A_1, x_2 : A_2)) \to \rho_1$ and $q(a(x_1 : A'_1, x_2 : A'_2)) \to \rho_2$ such that $L(A_1) \cap L(A'_1) \neq \emptyset$ and $L(A_2) \cap L(A'_2) \neq \emptyset$. It is known that the class of transformations induced by LDTT^R is a superclass of those by LDTT.

3.3. Schema k-Secrecy Problem

Definition 3.3.1 (k-Secrecy)**.** Given a database schema A_G, a database instance t, an authorized query T_a and an unauthorized query T_s where $Dom(T_a) \supseteq L(A_G)$ and $Dom(T_s) \supseteq L(A_G)$. We say that (t, T_a, T_s) is *k-secret* if $|Y(t)| \geq k$ where $Y(t) = \{T_s(t') \mid t' \in L(A_G), T_a(t') = T_a(t)\}$. In particular, (t, T_a, T_s) is *∞-secret* if $Y(t)$ is infinite. If T_a and T_s are clear from context, we simply say t is *$k(\infty)$-secret*.

In the above definition, $Y(t)$ represents the candidates of secret information. We assume that attackers know the schema, the definition of both authorized and unauthorized queries as well as the results of authorized queries applied to t, but do not know the instance t and the secret information $T_s(t)$. Note that $Y(t)$ can be represented as $Y(t) = T_s(T_a^{-1}(T_a(t)) \cap L(A_G))$. An attacker uses the available information from the result of authorized queries $T_a(t)$ to compute the inverse image $T_a^{-1}(T_a(t))$, then extracts from $T_a^{-1}(T_a(t))$ the trees that conform to the given schema A_G. After that, the attacker executes the unauthorized query T_s in order to obtain the candidate secret information $Y(t)$ of the database. $Y(t)$ is the set consisting of all candidates that the attacker cannot distinguish from the secret information $T_s(t)$.

Note that a database instance always satisfies 1-secrecy and hence when we refer to k-secrecy, we implicitly assume that $k \geq 2$.

Definition 3.3.2 (Schema k-Secrecy)**.** Given a database schema A_G, an authorized query T_a, and an unauthorized query T_s, (A_G, T_a, T_s) is *schema k-secret* if

every instance $t \in L(A_G)$ is k-secret.

Schema k-secrecy (or schema ∞-secrecy) problem, abbreviated as k-SS (or ∞-SS), asks whether, given a database schema A_G, an authorized query T_a, and an unauthorized query T_s, (A_G, T_a, T_s) is schema k-secret (or schema ∞-secret). In this study, we assume the number of authorized queries is one and leave the extension to multiple authorized queries as future work (see Section 3.6.1).

Example 3.3.3. Consider TA A_G and transducers T_a^1, T_a^2, T_s^1 and T_s^2 defined as follows:

- $\Sigma = \{a, \tilde{a}, s, \#\}$, $Q = \{q_0, q_1, q_\#\}$ and $A_G = (Q, \Sigma, \{q_0\}, \{q_0 \to a(q_1, q_\#), q_0 \to \tilde{a}(q_1, q_\#), q_1 \to s(q_1, q_\#), q_1 \to s(q_\#, q_\#), q_\# \to \#\})$. A_G accepts the trees which have the shapes shown in Figure 3.1.

- $T_a^i = (\{q\}, \Sigma, \Sigma, \{q\}, \delta_a^i)$ $(i = 1, 2)$ where $\delta_a^1 = \{q(a(x_1, x_2)) \to a(q(x_1), q(x_2)), q(\tilde{a}(x_1, x_2)) \to a(q(x_1), q(x_2)), q(s(x_1, x_2)) \to q(x_1), q(\#) \to \#\}$ and δ_a^2 is the set of rules obtained from δ_a^1 by replacing $q(\tilde{a}(x_1, x_2)) \to a(q(x_1), q(x_2))$ with $q(\tilde{a}(x_1, x_2)) \to \tilde{a}(q(x_1), q(x_2))$. Both T_a^1 and T_a^2 delete all s and T_a^1 replaces $\tilde{a}$ with a.

- T_s^1 is an LDTT that induces the identity map on $L(A_G)$.

- $T_s^2 = (\{q\}, \Sigma, \Sigma, \{q\}, \delta_s^2)$ where $\delta_s^2 = \{q(a(x_1, x_2)) \to a(\#, \#), q(\tilde{a}(x_1, x_2)) \to \tilde{a}(q(x_1), q(x_2)), q(s(x_1, x_2)) \to s(q(x_1), q(x_2)), q(\#) \to \#\}$. T_s^2 deletes all s only after it reads a.

For $m \geq 1$, let as^m and $\tilde{a}s^m$ be the trees of the shape shown in Figure 3.1 (a) and (b) respectively. Also let $as^+ = \{as^m \mid m \geq 1\}$ and $\tilde{a}s^+ = \{\tilde{a}s^m \mid m \geq 1\}$. A symbol a stands for $a(\#, \#)$ or $\{a(\#, \#)\}$ depending on the context. For each $t \in L(A_G)$, $T_a^1(t) = a$. Thus, $(T_a^1)^{-1} \circ T_a^1(t) \cap L(A_G) = L(A_G)$. $T_a^2(as^m) = a$ and $T_a^2(\tilde{a}s^m) = \tilde{a}$. Thus, $(T_a^2)^{-1} \circ T_a^2(as^m) \cap L(A_G) = as^+$ and $(T_a^2)^{-1} \circ T_a^2(\tilde{a}s^m) \cap L(A_G) = \tilde{a}s^+$. Since T_s^1 is the identity map, (A_G, T_a^1, T_s^1), (A_G, T_a^2, T_s^1) are schema ∞-secret. (A_G, T_a^1, T_s^2) is also schema ∞-secret because $T_s^2(L(A_G)) = a \cup \tilde{a}s^+$ while (A_G, T_a^2, T_s^2) is not schema ∞-secret because $T_s^2(as^+) = a$. See Table 3.1.

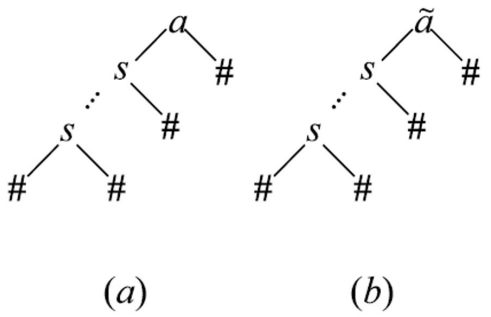

Figure 3.1. Trees accepted by A_G.

Table 3.1. Schema ∞-secrecy for Example 3.3.3.

T_a^i, T_s^i	t	$T_a^i(t)$	$A(t)$	$Y(t)$	$\lvert Y(t)\rvert$	∞-secret
T_a^1, T_s^1	$\begin{matrix} as^m \\ \tilde{a}s^m \end{matrix}$	a	$L(A_G)$	$L(A_G)$	∞	yes
T_a^2, T_s^1	$\begin{matrix} as^m \\ \tilde{a}s^m \end{matrix}$	$\begin{matrix} a \\ \tilde{a} \end{matrix}$	$\begin{matrix} as^+ \\ \tilde{a}s^+ \end{matrix}$	$\begin{matrix} as^+ \\ \tilde{a}s^+ \end{matrix}$	∞	yes
T_a^1, T_s^2	$\begin{matrix} as^m \\ \tilde{a}s^m \end{matrix}$	a	$L(A_G)$	$a \cup \tilde{a}s^+$	∞	yes
T_a^2, T_s^2	$\begin{matrix} as^m \\ \tilde{a}s^m \end{matrix}$	$\begin{matrix} a \\ \tilde{a} \end{matrix}$	$\begin{matrix} as^+ \\ \tilde{a}s^+ \end{matrix}$	$\begin{matrix} a \\ \tilde{a}s^+ \end{matrix}$	$\begin{matrix} 1 \\ \infty \end{matrix}$	no

We let $A(t) = (T_a^i)^{-1} \circ T_a^i(t) \cap L(A_G)$ and $Y(t) = T_s^i((T_a^i)^{-1} \circ T_a^i(t) \cap L(A_G))$. Note that $L(A_G) = as^+ \cup \tilde{a}s^+$.

3.4. Undecidability Result

We show that k-SS is undecidable for any finite $k > 1$, even if queries are restricted to deterministic rational transducers on words (DRTW) [5] where a rule has the shape $p(ax) \to \alpha q(x)$ (a is an input symbol, x is a variable denoting the remaining input string, p, q are states and α is a string of output symbols). DRTW can be regarded as a proper subclass of LDTT.

Theorem 3.4.1. *k-SS is undecidable for any integer $k > 1$, where queries are represented by DRTW.*

Proof. We show the undecidability of 2-SS by reduction from PCP. The undecidability of k-SS for $k > 2$ can be shown in almost the same way. Here, we use a regular expression to represent a schema and DRTW to represent queries. The class of languages represented by regular expressions, when regarded as tree languages, is a proper subclass of regular tree languages.

Consider any instance (W, U) of PCP, where $W = \langle w_1, w_2, \ldots, w_n \rangle$ and $U = \langle u_1, u_2, \ldots, u_n \rangle$ are both n-tuples of words over an alphabet Σ. Let $[n] = \{1, \ldots, n\}$ and we assume that $\Sigma \cap [n] = \emptyset$. We construct the following regular expression A_G over $\Sigma \cup [n]$ as a schema:

$$A_G = (1w_1 \mid 2w_2 \mid \ldots \mid nw_n)^+ \mid (1u_1 \mid 2u_2 \mid \ldots \mid nu_n)^+.$$

For example, let $W = \langle abc, d \rangle$ and $U = \langle a, bcd \rangle$, then we construct $A_G = (1abc \mid 2d)^+ \mid (1a \mid 2bcd)^+$. Let $L(A_G)$ be the language represented by A_G. Next, let T_a be the authorized query that extracts only symbols in $[n]$ from an input word, and let T_s be the unauthorized query that extracts only symbols in Σ from an input word. For example, given $t = 1abc2d$ as an input word, $T_a(t) = 12$ and $T_s(t) = abcd$. Both T_a and T_s can be defined by DRTW with one state.

We now show that $|T_s(T_a^{-1}(T_a(t)) \cap L(A_G))| \geq 2$ for any $t \in L(A_G)$ if and only if (W, U) has no solution. Take any $t = i_1 w_{i_1} i_2 w_{i_2} \cdots i_m w_{i_m} \in L(A_G)$ (or $t = i_1 u_{i_1} i_2 u_{i_2} \cdots i_m u_{i_m} \in L(A_G)$), and then it holds that $T_a^{-1}(T_a(t)) \cap L(A_G) = \{i_1 w_{i_1} i_2 w_{i_2} \cdots i_m w_{i_m},\ i_1 u_{i_1} i_2 u_{i_2} \cdots i_m u_{i_m}\}$. Thus,

$$Y(t) = T_s(T_a^{-1}(T_a(t)) \cap L(A_G)) = \{w_{i_1} w_{i_2} \cdots w_{i_m},\ u_{i_1} u_{i_2} \cdots u_{i_m}\}.$$

This situation is depicted in Figure 3.2. If the instance (W, U) of PCP has a solution $i_1 i_2 \cdots i_m$, then $|Y(t)| = 1$ where $t = i_1 w_{i_1} i_2 w_{i_2} \cdots i_m w_{i_m}$ or $t = i_1 u_{i_1} i_2 u_{i_2} \cdots i_m u_{i_m}$ because $w_{i_1} w_{i_2} \cdots w_{i_m} = u_{i_1} u_{i_2} \cdots u_{i_m}$. Conversely, if the instance (W, U) of PCP has no solution, then $|Y(t)| = 2$ for any $t \in L(A_G)$ because $w_{i_1} w_{i_2} \cdots w_{i_m} \neq u_{i_1} u_{i_2} \cdots u_{i_m}$ for any $i_1 i_2 \cdots i_m \in [n]^+$. Thus, we have proved that 2-SS is undecidable. □

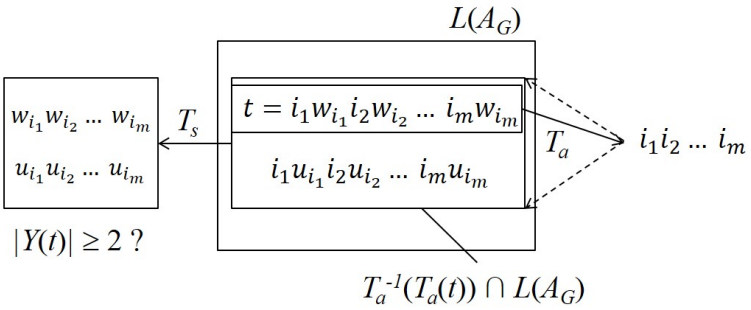

Figure 3.2. Reduction from PCP to 2-SS.

3.5. Decidability Results on ∞-SS

In this section, we first give an overview of our decision algorithm for ∞-SS in Section 3.5.1, and the details of the algorithm where queries are represented by LDTT in Section 3.5.2. Then, we prove the correctness of the algorithm in Section 3.5.3, and analyze the time complexity in Section 3.5.4. The results are extended to LDTT^R in Section 3.5.5.

3.5.1 Overview of Decision Algorithm

For simplicity, we consider only authorized/unauthorized queries but not schemas here. More precisely, we fix a schema to be a TA which accepts any tree. We assume that a schema is arbitrarily given as a TA when we give the detail of our algorithm in Section 3.5.2.

Consider a pair $(t, T(t))$ of input and output trees of an LDTT T. For each node v of $t \in Dom(T)$, v either is transformed to some nonempty part of $T(t)$ by applying some rule of T to v, or does not correspond to any part of $T(t)$, that is, v is *deleted* by T. For example, let T be the LDTT which has the transduction rules listed in Figure 3.3. The left tree is transformed into the right tree in Figure 3.3 by T. The nodes of the left tree in the circle-dot-lines correspond to the parts of the right tree according to the applied rules of T, respectively. On the other hand, the nodes in the rectangle-dot-line are deleted by T. The subtree $c(\#, \#)$ is deleted because $q_1(b(x_1, x_2)) \to \#$ is applied to the parent node b and thus the subtrees at x_1 and x_2 are ignored (or not touched). Note that the node b

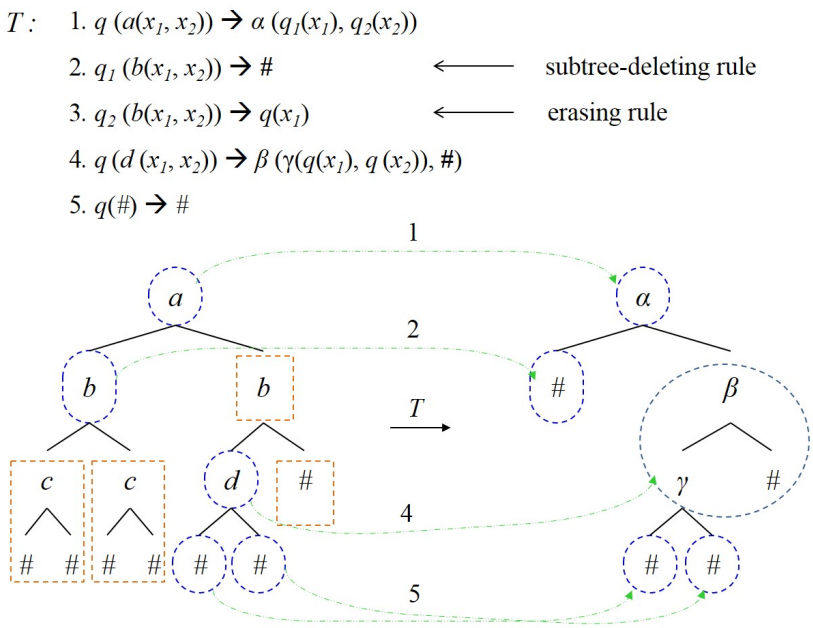

Figure 3.3. Deleted and undeleted nodes by T.

is not deleted but transformed to #. We call a rule such that a variable in its left hand side does not occur in its right hand side, like $q_1(b(x_1, x_2)) \to \#$, a *subtree-deleting* rule. The node b which is the right child of the root is deleted by T because $q_2(b(x_1, x_2)) \to q(x_1)$ is applied to the node and thus no symbol is output by this rule. We call a rule without output symbols in the right hand an *erasing* rule. Note that any erasing rule is also a subtree-deleting rule because only one variable can occur in its right-hand side of the erasing rule.

We intend to show a decision algorithm for ∞-secrecy of (A_G, T_a, T_s) which will be explained later. The unambiguous TA A_a can be constructed from a query T_a as follows:

- A_a recognizes the domain of T_a.
- Each state of A_a is either a rule ρ of T_a or a special state $\perp$.
- For any input tree t and any node v of t, if the unique run of A_a on t assigns a rule ρ of T_a, then T_a applies the rule ρ at v when t is transformed by T_a;

if the run assigns $\perp$ to v, then T_a deletes v by applying a subtree-deleting rule at some proper ancestor of v.

For an input tree $t \in Dom(T_a)$, in order to distinguish whether a node is deleted by T_a, we just look at the state assigned to the node of t by A_a. That is, $\perp$ or ρ where ρ is an erasing rule is assigned to the node if and only if the node is deleted by T_a. We call such states *deleting states* of A_a. Similarly, let A_s denote the unambiguous TA constructed from T_s in the same way as A_a.

Our algorithm consists of four steps:

1. Construct tree automata A_a and A_s from T_a and T_s s.t $L(A_a) = Dom(T_a)$ and $L(A_s) = Dom(T_s)$.

2. Construct $A_I = A_a \times A_s \times A_G$.

3. Construct tree automata A_{inf} and A_{fin} from A_I such that

   - $L(A_{fin}) \cup L(A_{inf}) = L(A_I)$, $L(A_{fin}) \cap L(A_{inf}) = \emptyset$ and
   - $T_a(L(A_{fin})) \subseteq T_a(L(A_{inf}))$ iff (A_G, T_a, T_s) is ∞-secret

   by identifying the states of A_I which can occur recursively and contain deleting states of A_a and non-deleting states of A_s as their components.

4. Decide whether $T_a(L(A_{fin})) \subseteq T_a(L(A_{inf}))$.

3.5.2 Detailed Construction

Let $A_G = (Q, \Sigma, Q_0, \delta)$ be a TA as an XML schema and let $T_a = (Q_a, \Sigma, \Delta, \{q_a^0\}, \delta_a)$ and $T_s = (Q_s, \Sigma, \Delta, \{q_s^0\}, \delta_s)$ be LDTT as authorized and unauthorized queries, respectively. Our algorithm works as follows:

Step 1. Construct the following two TAs A_a and A_s from T_a and T_s:

- $A_a = (\delta_a \cup \{\perp\}, \Sigma, \delta_{a0}, \delta'_a)$ where $\perp \notin Q_a$, δ_{a0} is the set of rules in δ_a of which left hand side contains q_a^0 and δ'_a is the smallest set such that

  - for each $\rho = q(\sigma(x_1, x_2)) \to C[q_1(x_1), q_2(x_2)] \in \delta_a$ with $q, q_1, q_2 \in Q_a$, $\sigma \in \Sigma$ and $C \in \mathcal{C}(\Delta, \mathcal{X}_2)$,

$$\rho \to \sigma(\tilde{q}_1, \tilde{q}_2) \in \delta'_a$$

where for each $i \in \{1, 2\}$, if the right hand side of ρ (abbreviated as RHS(ρ)) contains x_i then $\tilde{q}_i = \rho_i$ for any $\rho_i \in \delta_a$ of which left hand side contains q_i; otherwise $\tilde{q}_i = \bot$.

- for each $\rho = q(\#) \to u$ where $q \in Q_a$ and $u \in \mathcal{T}_\Delta$, $\rho \to \# \in \delta'_a$.
- for each $\sigma \in \Sigma$, $\bot \to \sigma(\bot, \bot) \in \delta'_a$.
- $\bot \to \# \in \delta'_a$.

If the variable x_i disappears from RHS(ρ), we use the dummy state $\bot$ instead of q_i so that A_a goes through the subtree substituted into x_i by the rule $\bot \to \sigma(\bot, \bot)$.

- $A_s = (\delta_s \cup \{\bot\}, \Sigma, \delta_{s0}, \delta'_s)$ is defined from T_s in the same way as A_a. We can see that A_a and A_s constructed from T_a and T_s as above satisfy the three conditions listed in Section 3.5.1. Note that A_a and A_s are unambiguous TAs because T_a and T_s are top-down deterministic and thus it is uniquely determined which rule is applied to each node.

Step 2. Construct the product TA A_I of A_a, A_s, and A_G. More specifically, construct the following tree automaton $A_I = (Q'_a \times Q'_s \times Q, \Sigma, Q^0_a \times Q^0_s \times Q_0, \delta_I)$ from $A_a = (Q'_a, \Sigma, Q^0_a, \delta'_a)$, $A_s = (Q'_s, \Sigma, Q^0_s, \delta'_s)$, and $A_G = (Q, \Sigma, Q_0, \delta)$: $(q_a, q_s, q) \to \sigma((q^1_a, q^1_s, q^1), (q^2_a, q^2_s, q^2)) \in \delta_I$ if and only if $q_a \to \sigma(q^1_a, q^2_a) \in \delta'_a$, $q_s \to \sigma(q^1_s, q^2_s) \in \delta'_s$, and $q \to \sigma(q^1, q^2) \in \delta$. Note that $L(A_G) = L(A_I)$ because $Dom(T_a) \supseteq L(A_G)$ and $Dom(T_s) \supseteq L(A_G)$. Also for every $t \in L(A_I)$, there are unique runs r_a, r_s of A_a, A_s respectively, on t because A_a, A_s are unambiguous; r is a run of A_G on t if and only if r' is a run of A_I on t such that $r'(v) = (r_a(v), r_s(v), r(v))$ for every node $v \in V(t)$.

Step 3. Let $Q_I = Q'_a \times Q'_s \times Q$ be the set of states of A_I. Compute two subsets $\mathcal{Q}^D_a$ and $\mathcal{Q}^D_s$ of Q_I as follows:

$$\begin{aligned}
\mathcal{Q}^D_a &= \{(q_a, q_s, q) \in Q_I \mid q_a = \bot \text{ or } q_a = \rho \\
&\qquad \text{where } \rho \text{ is an erasing rule of } T_a\}, \\
\mathcal{Q}^D_s &= \{(q_a, q_s, q) \in Q_I \mid q_s = \bot \text{ or } q_s = \rho \\
&\qquad \text{where } \rho \text{ is an erasing rule of } T_s\}.
\end{aligned}$$

A_I assigns a state $(q_a, q_s, q) \in \mathcal{Q}_a^D$ (rsp. $\mathcal{Q}_s^D$) to a node v if and only if T_a (rsp. T_s) deletes v by a subtree-deleting rule.

Step 4. Compute Q' consisting of $q_I \in Q_I$ such that q_I is recursive through a nonempty path of the dependency graph G_{A_I} such that all the states in the path are in $\mathcal{Q}_a^D$ and some state in the path is not in $\mathcal{Q}_s^D$. Let $\mathcal{Q}_a^D = \{1, 2, ..., n\}$ where $n = |\mathcal{Q}_a^D|$. We can compute Q' by dynamic programming using $\mathcal{D}_k[i, j]$ $(k, i, j \in \mathcal{Q}_a^D)$. We intend $\mathcal{D}_k[i, j] = 0$ if there is no path from i to j via $1, 2, ..., k$. $\mathcal{D}_k[i, j] = 1$ if there is a path from i to j via $1, 2, ..., k$ and every node in such a path belongs to $\mathcal{Q}_s^D$. $\mathcal{D}_k[i, j] \geq 2$ if there is a path from i to j via $1, 2, ..., k$ such that at least one node in the path does not belong to $\mathcal{Q}_s^D$.

$\mathcal{D}_0[i, j]$ can be defined as follows: for each $i, j \in \mathcal{Q}_a^D$

- if $\exists\ k \in \mathcal{Q}_a^D\ \exists \sigma \in \Sigma$: $i \to \sigma(j, k) \in \delta_I$ or $i \to \sigma(k, j) \in \delta_I$ then
  - if $i, j \in \mathcal{Q}_s^D$ then $\mathcal{D}_0[i, j] := 1$
  - else $\mathcal{D}_0[i, j] := 2$
- else $\mathcal{D}_0[i, j] := 0$.

We compute $\mathcal{D}_k[i, j] (1 \leq k \leq n)$ as follows: $\mathcal{D}_k[i, j] = \max\{\mathcal{D}_{k-1}[i, k] \times \mathcal{D}_{k-1}[k, j], \mathcal{D}_{k-1}[i, j]\}$. Then Q' can be obtained as $\{i \mid \mathcal{D}_n[i, i] \geq 2\}$.

Step 5. Construct A_{inf} such that $t \in L(A_{inf})$ if and only if there is a run r_t^I of A_I on t that assigns a state in Q' to at least one node of t as follows: $A_{inf} = (Q_I \times \{0, 1\}, \Sigma, \{q_a^0\} \times \{q_s^0\} \times Q_0 \times \{1\}, \delta_I'')$ where δ_I'' is the smallest set such that

- for each $q \to \sigma(q_1, q_2) \in \delta_I$, if $q \in Q'$ then $(q, 1) \to \sigma((q_1, i), (q_2, j)) \in \delta_I''$ for each $i, j \in \{0, 1\}$; otherwise, $(q, i||j) \to \sigma((q_1, i), (q_2, j)) \in \delta_I''$ for each $i, j \in \{0, 1\}$, where $||$ is the boolean operator OR, and
- for each $q \to \# \in \delta_I$, if $q \in Q'$ then $(q, 1) \to \# \in \delta_I''$; otherwise $(q, 0) \to \# \in \delta_I''$.

The last component $\{0, 1\}$ of each state is for checking whether at least one state in Q' appears in a run.

Step 6. Construct A_{fin} such that $L(A_{fin}) = L(A_I) - L(A_{inf})$. More concretely, construct a complement TA A^c_{inf} of A_{inf} and then construct an intersection TA of A_I and A^c_{inf}. Note that TAs are closed under Boolean operations [5].

Step 7. Decide whether $T_a(L(A_{fin})) \subseteq T_a(L(A_{inf}))$. If $T_a(L(A_{fin})) \subseteq T_a(L(A_{inf}))$ then output "schema ∞-secret"; Otherwise, output "not schema ∞-secret."

Note that any LDTT T preserves regularity, that is, for any regular tree language L, $T(L)$ is also a regular language, and a TA which recognizes $T(L)$ can be constructed. Also, the inclusion problem is decidable for regular tree languages.

Example 3.5.1. Using the above algorithm, we can decide ∞-secrecy for Example 3.3.3 as follows:

Table 3.2. The execution results of the algorithm for Example 3.3.3.

T^i_a, T^i_s	$L(A_{inf})$	$L(A_{fin})$	$T^i_a(L(A_{inf}))$	$T^i_a(L(A_{fin}))$	H
T^1_a, T^1_s	$L(A_G)$	$\emptyset$	a	$\emptyset$	yes
T^2_a, T^1_s	$L(A_G)$	$\emptyset$	$a \cup \tilde{a}$	$\emptyset$	yes
T^1_a, T^2_s	$\tilde{a}s^+$	as^+	a	a	yes
T^2_a, T^2_s	$\tilde{a}s^+$	as^+	$\tilde{a}$	a	no

We let $H = T^i_a(L(A_{fin})) \subseteq T^i_a(L(A_{inf}))$.

Consider the third case (A_G, T^1_a, T^2_s) in Table 3.2. We assign a label to each rule of T^1_a and T^2_s as $\delta^1_a = \{\rho^1_a{:}q(a(x_1, x_2)) \to a(q(x_1), q(x_2)), \rho^2_a{:}q(\tilde{a}(x_1, x_2)) \to a(q(x_1), q(x_2)), \rho^3_a{:}q(s(x_1, x_2)) \to q(x_1), \rho^4_a{:}q(\#) \to \#\}$. $\delta^2_s = \{\rho^1_s{:}q(a(x_1, x_2)) \to a(\#, \#), \rho^2_s{:}q(\tilde{a}(x_1, x_2)) \to \tilde{a}(q(x_1), q(x_2)), \rho^3_s{:}q(s(x_1, x_2)) \to s(q(x_1), q(x_2)), \rho^4_s{:}q(\#) \to \#\}$. We can construct $A^1_a, A^2_s, A_I, A_{inf}$ and obtain Q' as follows. Note that useless states and rules are removed at (Step 2) and (Step 5).

- (Step 1) $A^1_a = (Q^{1\prime}_a, \Sigma, \delta^1_a, \delta^{1\prime}_a)$ where $Q^{1\prime}_a = \delta^1_a \cup \{\bot\}$, $\delta^{1\prime}_a = \{\rho^1_a \to a(\rho^i_a, \rho^j_a), \rho^2_a \to \tilde{a}(\rho^i_a, \rho^j_a), \rho^3_a \to s(\rho^i_a, \bot), \rho^4_a \to \#, \bot \to \sigma(\bot, \bot), \bot \to \#\}$ with $1 \le i \le 4$, $1 \le j \le 4$ and $\sigma \in \{a, \tilde{a}, s\}$. $A^2_s = (Q^{2\prime}_s, \Sigma, \delta^2_s, \delta^{2\prime}_s)$ where $Q^{2\prime}_s = \delta^2_s \cup \{\bot\}$, $\delta^{2\prime}_s = \{\rho^1_s \to a(\bot, \bot), \rho^2_s \to \tilde{a}(\rho^i_s, \rho^j_s), \rho^3_s \to s(\rho^i_s, \rho^j_s), \rho^4_s \to \#, \bot \to \sigma(\bot, \bot), \bot \to \#\}$ with $1 \le i \le 4$, $1 \le j \le 4$ and $\sigma \in \{a, \tilde{a}, s\}$.

- (Step 2) $A_I = (Q_a^{1\prime} \times Q_s^{2\prime} \times Q, \Sigma, \{(\rho_a^1, \rho_s^1, q_0), (\rho_a^2, \rho_s^2, q_0)\},$
$\delta_I)$ where $\delta_I = \{(\rho_a^1, \rho_s^1, q_0) \to a((\rho_a^3, \perp, q_1), (\rho_a^4, \perp, q_\#)),$
$(\rho_a^2, \rho_s^2, q_0) \to \tilde{a}((\rho_a^3, \rho_s^3, q_1), (\rho_a^4, \rho_s^4, q_\#)),$
$(\rho_a^3, \rho_s^3, q_1) \to s((\rho_a^3, \rho_s^3, q_1), (\perp, \rho_s^4, q_\#)),$
$(\rho_a^3, \rho_s^3, q_1) \to s((\rho_a^4, \rho_s^4, q_\#), (\perp, \rho_s^4, q_\#)),$
$(\rho_a^3, \perp, q_1) \to s((\rho_a^3, \perp, q_1), (\perp, \perp, q_\#)),$
$(\rho_a^3, \perp, q_1) \to s((\rho_a^4, \perp, q_\#), (\perp, \perp, q_\#)),$
$(\rho_a^4, \rho_s^4, q_\#) \to \#$, $(\rho_a^4, \perp, q_\#) \to \#$, $(\perp, \rho_s^4, q_\#) \to \#$, $(\perp, \perp, q_\#) \to \#\}$.

- (Step 3) $\mathcal{Q}_a^D$ is the subset of states of which first component is ρ_a^3 or $\perp$. $\mathcal{Q}_s^D$ is the subset of states of which second component is $\perp$.

- (Step 4) The dependency graph G_{A_I} of A_I is shown in Figure 3.4. $Q' = \{(\rho_a^3, \rho_s^3, q_1)\}$.

- (Step 5) $A_{inf} = (Q_I \times \{0, 1\}, \Sigma, \{((\rho_a^2, \rho_s^2, q_0), 1)\}, \delta_I'')$ where $\delta_I{}'' = \{((\rho_a^2, \rho_s^2, q_0), 1) \to \tilde{a}(((\rho_a^3, \rho_s^3, q_1), 1), ((\rho_a^4,$
$\rho_s^4, q_\#), 0)), ((\rho_a^3, \rho_s^3, q_1), 1) \to s(((\rho_a^3, \rho_s^3, q_1), 1), ((\perp, \rho_s^4,$
$q_\#), 0)), ((\rho_a^3, \rho_s^3, q_1), 1) \to s(((\rho_a^4, \rho_s^4, q_\#), 0), ((\perp, \rho_s^4, q_\#),$
$0)), ((\rho_a^4, \rho_s^4, q_\#), 0) \to \#$, $((\rho_a^4, \perp, q_\#), 0) \to \#$, $((\perp, \rho_s^4, q_\#), 0) \to \#\}$.

- (Step 6) A_{fin} is a TA such that $L(A_{fin}) = L(A_I) - L(A_{inf})$.

- (Step 7) We have $L(A_{inf}) = \tilde{a}s^+$, $L(A_{fin}) = L(A_G) - L(A_{inf}) = as^+$. $T_a^1(L(A_{fin})) \subseteq T_a^1(L(A_{inf}))$ holds and (A_G, T_a^1, T_s^2) is schema ∞-secret because $T_a^1(L(A_{fin})) = a$ and $T_a^1(L(A_{inf})) = a$. See Table 3.2.

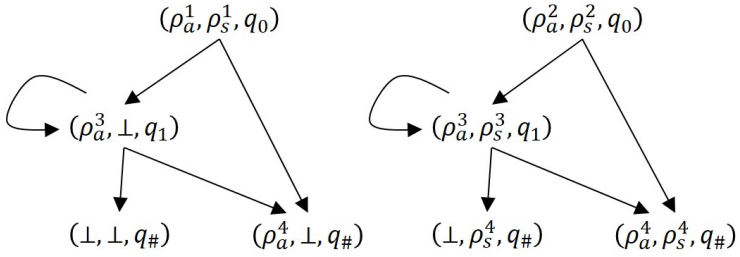

Figure 3.4. The dependency graph G_{A_I}.

3.5.3 Correctness

We give several lemmas for the proof of the correctness.

Lemma 3.5.2. *Let t be an arbitrary tree in $L(A_I)$. For any node v_1 and any descendant v_2 of v_1 in t such that*

- *there is a run r_t^I of A_I on t such that $r_t^I(v_1) = r_t^I(v_2)$ and*
- *every node v that appears in the path from v_1 to v_2 (inclusive) is mapped to a state in $\mathcal{Q}_a^D$ (rsp. in $\mathcal{Q}_s^D$) by the run r_t^I,*

$T_a(t) = T_a(t[v_1 \leftarrow t/v_2])$ (rsp. $T_s(t) = T_s(t[v_1 \leftarrow t/v_2])$) (see Figure 3.5).

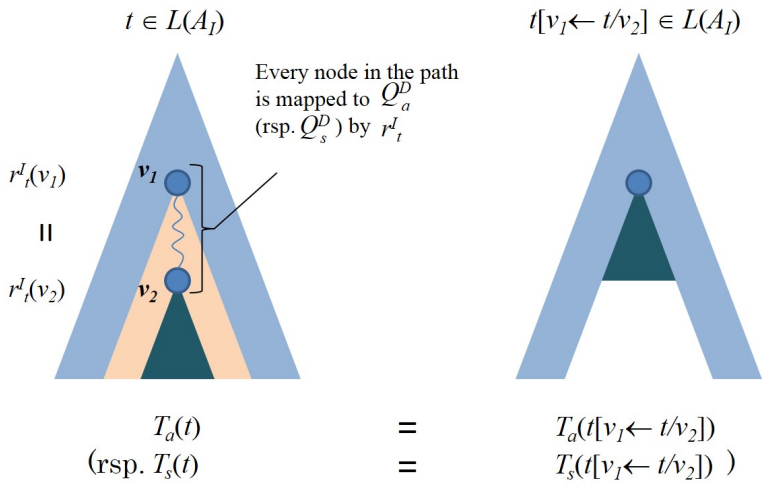

Figure 3.5. Lemma 3.5.2.

Proof. Let t be an arbitrary tree in $L(A_I)$ and let v be an arbitrary node in t. Let r_t^I be an arbitrary run of A_I on t. By definition of A_a, A_s, and A_I, we have the following two facts:

- If $r_t^I(v) = (\bot, q_s, q) \in \mathcal{Q}_a^D$ then for every descendant v' of v, $r_t^I(v') = (\bot, q_s', q') \in \mathcal{Q}_a^D$ for some $q_s' \in Q_s$ and $q' \in Q$.
- If $r_t^I(v) = (q_a, q_s, q) \in \mathcal{Q}_a^D$ where $q_a \neq \bot$ then for either one node v' of the two children of v, $r_t^I(v') = (\bot, q_s', q') \in \mathcal{Q}_a^D$ for some $q_s' \in Q_s$ and $q' \in Q$.

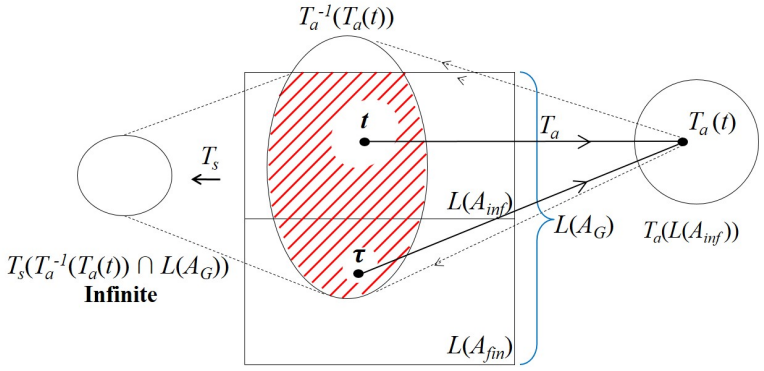

Figure 3.6. Lemma 3.5.3.

Let v_1 be an arbitrary node in t, and let v_2 be an arbitrary descendant of v_1. We assume that $r_t^I(v_1) = r_t^I(v_2) \in \mathcal{Q}_a^D$ and that $r_t^I(v) \in \mathcal{Q}_a^D$ for each v that appears in the path from v_1 to v_2. By the facts we stated above, for each node v that is a descendant of v_1 but not a descendant of v_2, $r_t^I(v) \in \mathcal{Q}_a^D$. Since nodes mapped to states in $\mathcal{Q}_a^D$ by r_t^I are deleted by T_a, it holds that $T_a(t) = T_a(t[v_1 \leftarrow t/v_2])$.

The case for T_s and $\mathcal{Q}_s^D$ can be shown in the same way.

□

Lemma 3.5.3. *For every $\tau \in L(A_G)$, if $T_a(\tau) \in T_a(L(A_{inf}))$ then $T_s(T_a^{-1}(T_a(\tau)) \cap L(A_G))$ is infinite.*

Proof. Assume that $T_a(\tau) \in T_a(L(A_{inf}))$, then there is $t \in L(A_{inf})$ such that $T_a(t) = T_a(\tau)$ (see Figure 3.6). Note that if $\tau \in L(A_{inf})$ we just let $t = \tau$. By the definition of A_{inf}, there is a run r_t^I of A_I on t that assigns a state in Q' to at least one node of t. Let v be an arbitrary node such that $r_t^I(v) \in Q'$. By the definition of Q', there are another tree $t' \in L(A_{inf})$ having a node v_1 and its proper descendant v_2, and a run $r_{t'}^I$ of A_I on t' such that

- $t'[v_1 \leftarrow t'/v_2] = t$ and $t/v = t'/v_2$,
- $r_{t'}^I(v_1) = r_{t'}^I(v_2) = r_t^I(v) \in Q'$, and
- all the states in the path from v_1 and v_2 are in $\mathcal{Q}_a^D$ and some state in the path is not in $\mathcal{Q}_s^D$.

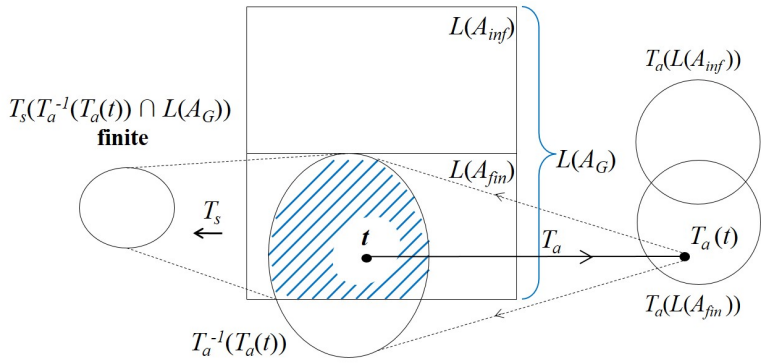

Figure 3.7. Lemma 3.5.4.

By Lemma 3.5.2, $T_a(t') = T_a(t)$. On the other hand, $T_s(t') \neq T_s(t)$ because there is some state $q_s \notin \mathcal{Q}_s^D$ between v_1 and v_2 in t' and thus at least one more node labeled with an output symbol in Δ is emitted by T_s, that is, $|T_s(t')| \geq |T_s(t)|+1$. Let t^k and t^{k+1} be the trees in $L(A_{inf})$ obtained from t by pumping at v as above k and $k+1$ times, respectively, then $T_a(t^k) = T_a(t^{k+1})$ and $T_s(t^k) \neq T_s(t^{k+1})$. Thus, $T_s(T_a^{-1}(T_a(t)) \cap L(A_G))$ is infinite. □

Lemma 3.5.4. *For every $t \in L(A_{fin})$, if $T_a(t) \notin T_a(L(A_{inf}))$ then $T_s(T_a^{-1}(T_a(t)) \cap L(A_G))$ is finite (see Figure 3.7).*

Proof. Let t be an arbitrary tree in $L(A_{fin})$ and let r_t^I be an arbitrary run of A_I on t. For any node v_1 and any descendant v_2 of v_1 in t such that

- $r_t^I(v_1) = r_t^I(v_2)$ and
- every node v that appears in the path from v_1 to v_2 is mapped to a state in $\mathcal{Q}_a^D$ by r_t^I,

it holds not only that $T_a(t) = T_a(t[v_1 \leftarrow t/v_2])$ but also that $T_s(t) = T_s(t[v_1 \leftarrow t/v_2])$. The reason is as follows: let $\bar{q} = r_t^I(v_1) = r_t^I(v_2)$, then $\bar{q}$ is recursive through a path in t such that all of the states in the path are in $\mathcal{Q}_a^D$. By the fact that $L(A_{fin}) = L(A_I) - L(A_{inf})$ and the construction of A_{inf}, there is no run of A_I on t that assigns a state in Q' to some node of t (see Step 5 in Section 3.5.2). Hence, we must have that $r_t^I(v_1) = r_t^I(v_2) \in \mathcal{Q}_s^D$ and $r_t^I(v) \in \mathcal{Q}_s^D$ for each v which appears in the path from v_1 to v_2. Thus, by Lemma 3.5.2, $T_s(t) = T_s(t[v_1 \leftarrow t/v_2])$.

Now consider an arbitrary tree $t \in L(A_{fin})$ such that $T_a(t) \notin T_a(L(A_{inf}))$, and let $t_0 = T_a(t)$. For any $t' \in T_a^{-1}(t_0)$ and any run $r^I_{t'}$ of A_I on t', $|\{v \in V(t') \mid r^I_{t'}(v) \notin \mathcal{Q}^D_a\}| \leq |t_0|$ because a node labeled with an output symbol in Δ is emitted every time a rule that contains $q \notin \mathcal{Q}^D_a$ in its left-hand side is applied. Thus, in any path of t', there are at most $|t_0|$ nodes to which A_I assigns states not in $\mathcal{Q}^D_a$. Let l be the number of such nodes. In addition, in the path, there are $l+1$ intervals separated by the l nodes, and A_I assigns states in $\mathcal{Q}^D_a$ to all nodes in the intervals. Hence, using the fact we have stated in the beginning of this proof, we can say that for each $t' \in T_a^{-1}(t_0)$, there is a tree t'' such that $T_s(t'') = T_s(t')$ and the height of t'' is at most $|t_0| + (|t_0|+1)|\mathcal{Q}^D_a|$. Thus, there is a subset L_f of the set of all trees of height at most $|t_0| + (|t_0|+1)|\mathcal{Q}^D_a|$ such that $T_s(L_f) = T_s(T_a^{-1}(t_0))$. Since T_s is functional and L_f is finite, $T_s(T_a^{-1}(t_0) \cap L(A_G))$ is finite. □

We now give a lemma for correctness of our algorithm.

Lemma 3.5.5. *$T_a(L(A_{fin})) \subseteq T_a(L(A_{inf}))$ if and only if $T_s(T_a^{-1}(T_a(t)) \cap L(A_G))$ is infinite for every $t \in L(A_G)$.*

Proof. Assume that $T_a(L(A_{fin})) \subseteq T_a(L(A_{inf}))$. We have $T_a(t) \in T_a(L(A_{inf}))$ for every $t \in L(A_G)$. By Lemma 3.5.3, $T_s(T_a^{-1}(T_a(t)) \cap L(A_G))$ is infinite for every $t \in L(A_G)$.

On the other hand, assume that $T_a(L(A_{fin})) \subsetneq T_a(L(A_{inf}))$, and then there is some $t \in L(A_{fin}) \subseteq L(A_G)$ such that $T_a(t) \notin T_a(L(A_{inf}))$. Thus, by Lemma 3.5.4, $T_s(T_a^{-1}(T_a(t)) \cap L(A_G))$ is finite. □

By lemma 3.5.5, we obtain the following theorem.

Theorem 3.5.6. *∞-SS is decidable for LDTT.*

3.5.4 Time Complexity

Here, we show that ∞-SS for LDTT is EXPTIME-complete. First we show the time complexity of the proposed algorithm for ∞-SS where queries are represented by LDTT is in EXPTIME. Next we prove that ∞-SS for LDTT is EXPTIME-hard.

We estimate the time and space complexities of the proposed algorithm.

Step 1. Construct two TAs A_a and A_s from T_a and T_s: it takes $O(|T_a|^3)$ (rsp. $O(|T_s|^3)$) time to construct A_a (rsp. A_s), and its size is $O(|T_a|^3)$ (rsp. $O(|T_s|^3)$).

Step 2. Construct the product TA A_I from A_s, A_a and A_G: it takes $O(|A_a| \times |A_s| \times |A_G|)$ time, and its size is $O(|A_a| \times |A_s| \times |A_G|)$.

Step 3. Compute $\mathcal{Q}_a^D$ and $\mathcal{Q}_s^D$: it takes $O(|A_I| \times |A_a|)$ (rsp. $O(|A_I| \times |A_s|)$) time to construct $\mathcal{Q}_a^D$ (rsp. $\mathcal{Q}_s^D$), and the sizes of $\mathcal{Q}_a^D$ and $\mathcal{Q}_s^D$ are $O(|A_I|)$.

Step 4. Construct Q' from A_I, $\mathcal{Q}_a^D$, and $\mathcal{Q}_s^D$: construction of Q' takes $O(|A_I|^3)$ time, and its size is $O(|A_I|)$ because Q' is a subset of Q_I.

Step 5. Construct A_{inf} from A_I and Q': construction of A_{inf} takes $O(|A_I| \times |Q'|)$ time, and $|A_{inf}|$ is $O(|A_I|)$.

Step 6. Construct A_{fin} such that $L(A_{fin}) = L(A_I) - L(A_{inf})$: construction of a complement TA A_{inf}^c of A_{inf} takes exponential time and its size is exponential in $|A_{inf}|$ at the worst. Construction of an intersection TA A_{fin} of A_I and A_{inf}^c takes $O(|A_I| \times |A_{inf}^c|)$ time and its size is $O(|A_I| \times |A_{inf}^c|)$.

Step 7. Decide whether $T_a(L(A_{fin})) \subseteq T_a(L(A_{inf}))$: a TA B_{fin} recognizing $T_a(L(A_{fin}))$ can be constructed in $O(|T_a| \times |A_{fin}|)$ time. Similarly, a TA B_{inf} recognizing $T_a(L(A_{inf}))$ can be constructed in $O(|T_a| \times |A_{inf}|)$ time. Note that the size of B_{fin} is exponential in the sizes of T_a, T_s, and A_G but that of B_{inf} is polynomial. Thus, we can obtain a complement TA B_{inf}^c of B_{inf} of exponential size, and deciding whether $L(B_{fin}) \cap L(B_{inf}^c) = \emptyset$ is possible in $O(|B_{fin}| \times |B_{inf}^c|)$ time.

Therefore, the total time complexity of our algorithm is in EXPTIME.

Theorem 3.5.7. *∞-SS is EXPTIME-complete for LDTT.*

Proof. We showed ∞-SS is in EXPTIME solvable and thus we show ∞-SS is EXPTIME-hard by reduction from inclusion of regular tree languages, which is known to be EXPTIME-complete. The reduction uses Lemma 3.5.5.

Consider any two TAs $A_1 = (Q_1, \Sigma, Q_{10}, \delta_1)$ and $A_2 = (Q_2, \Sigma, Q_{20}, \delta_2)$. We construct A_G, T_a, and T_s such that $T_a(L(A_{fin})) = L(A_1)$ and $T_a(L(A_{inf})) =$

$L(A_2)$. Without loss of generality, neither A_1 nor A_2 accepts the tree having only $\#$, and Q_1 and Q_2 are disjoint. Let $\Sigma_{A_i} = \{a_{(q,q_1,q_2)} \mid q \to a(q_1, q_2) \in \delta_i\}(i = 1, 2)$. For $a_{(q,q_1,q_2)} \in \Sigma_{A_1} \cup \Sigma_{A_2}$, let $\pi(a_{(q,q_1,q_2)}) = a$. Note that Σ_{A_1} and Σ_{A_2} are disjoint because so are Q_1 and Q_2. We construct the following TA A_G and transducers T_a and T_s:

- $A_G = (Q_1 \cup Q_2 \cup \{q_0\}, \Sigma_{A_1} \cup \Sigma_{A_2} \cup \{s\}, Q_{10} \cup \{q_0\}, \delta_G)$ where q_0 is a new state, s is a new symbol, and δ_G is the smallest set such that
  - for each $a \in \Sigma$, if $q \to a(q_1, q_2) \in \delta_1 \cup \delta_2$ then $q \to a_{(q,q_1,q_2)}(q_1, q_2) \in \delta_G$,
  - for each $q_{20} \in Q_{20}$, $q_0 \to s(q_{20}, q_\#) \in \delta_G$,
  - $q_0 \to s(q_0, q_\#) \in \delta_G$, and
  - $q_\# \to \# \in \delta_G$.
- $T_a = (\{q_a\}, \Sigma_{A_1} \cup \Sigma_{A_2} \cup \{s\}, \Sigma, \{q_a\}, \delta_a)$ where $\delta_a = \{q_a(\tilde{a}(x_1, x_2)) \to a(q_a(x_1), q_a(x_2)) \mid \tilde{a} \in \Sigma_{A_1} \cup \Sigma_{A_2}, \pi(\tilde{a}) = a\} \cup \{q_a(s(x_1, x_2)) \to q_a(x_1), q_a(\#) \to \#\}$.
- $T_s = (\{q_s\}, \Sigma_{A_1} \cup \Sigma_{A_2} \cup \{s\}, \Sigma_{A_1} \cup \Sigma_{A_2} \cup \{s\}, \{q_s\}, \delta_s)$ where $\delta_s = \{q_s(\sigma(x_1, x_2)) \to \sigma(q_s(x_1), q_s(x_2)) \mid \sigma \in \Sigma_{A_1} \cup \Sigma_{A_2} \cup \{s\}\} \cup \{q_s(\#) \to \#\}$, which represents the identity transformation on trees over $\Sigma_{A_1} \cup \Sigma_{A_2} \cup \{s\}$.

Note that A_G is an unambiguous TA, and T_a and T_s are LDTTs. $L(A_G)$ consists of two disjoint sets: the set L_1 of trees over Σ_{A_1} corresponding to trees in $L(A_1)$ and the set L_2 of trees over $\Sigma_{A_2} \cup \{s\}$ corresponding to trees in $L(A_2)$. For $t \in L(A_1)$, the size of $T_a^{-1}(t) \cap L_1$ is at most the number of runs of A_1 on t. On the other hand, for $t \in L(A_2)$, there are infinitely many trees in L_2 that are transformed to t by T_a because trees in L_2 are allowed to have any number of s on the top and T_a deletes all the ss. From the above facts, we can see that A_{fin} and A_{inf} constructed by our decision algorithm in Section 3.5.2 satisfy that $T_a(L(A_{fin})) = L(A_1)$ and $T_a(L(A_{inf})) = L(A_2)$. Hence, from Lemma 4, $L(A_1) \subseteq L(A_2)$ if and only if (A_G, T_a, T_s) is ∞-secret. Thus, we obtain EXPTIME-hardness of ∞-SS. □

EXPTIME-hardness of LDBT can be shown in a similar way as above.

3.5.5 Extension to LDTT^R

We show that ∞-SS is decidable for LDTT^{R}s by revising Steps 1 and 3 of the algorithm in Section 3.5.2.

Step 1. Construct the following TA A_a from a given LDTT^R $T_a = (Q_a, \Sigma, \Delta, q_a^0, \delta_a)$:

Step 1.1 Let $\mathcal{A}$ be the set of all the TAs occurring as ranges (look-ahead) of variables in the rules of T_a. Let $A_{\rho,i}$ denote the TA of ranges of variable x_i in the rule ρ. For each $A_{\rho,i} \in \mathcal{A}$, construct a bottom-up deterministic [5] and complete TA $A'_{\rho,i}$ equivalent to $A_{\rho,i}$. Let $A'_{\rho,i} = (Q'_{\rho,i}, \Sigma, Q'_{0,\rho,i}, \delta_{\rho,i})$, and let $\mathcal{A}' = \{A'_{\rho,i} \mid A_{\rho,i} \in \mathcal{A}\}$.

Step 1.2 Construct a product TA $A^{rl} = (Q^{rl}, \Sigma, Q_0^{rl}, \delta^{rl})$ of all the TAs $A'_{\rho,i}$. For $\bar{q} \in Q^{rl}$, let $\pi_{\rho,i}(\bar{q})$ be the state of $A'_{\rho,i}$ occurring in $\bar{q}$.

Step 1.3 Construct $A_a = ((\delta_a \cup \{\bot\}) \times Q^{rl}, \Sigma, \delta_{a0} \times Q^{rl}, \delta')$ where δ_{a0} is the set of rules in δ_a of which left hand side contains q_a^0 and δ' is the smallest set such that

- for each $\rho = q(\sigma(x_1 : A_1, x_2 : A_2)) \to C[q_1(x_1), q_2(x_2)] \in \delta_a$ where $q_1, q_2, q \in Q_a$ and $\sigma \in \Sigma$,

$$(\rho, \bar{q}) \to \sigma((\tilde{q}_1, \bar{q}_1), (\tilde{q}_2, \bar{q}_2)) \in \delta'$$

  where $\bar{q} \to \sigma(\bar{q}_1, \bar{q}_2) \in \delta^{rl}$, for $i \in \{1, 2\}, \pi_{\rho,i}(\bar{q}_i) \in Q'_{0,\rho,i}$, and
  - if RHS(ρ) contains x_i then $\tilde{q}_i = \rho_i$ for any $\rho_i \in \delta_a$ of which left hand side contains q_i, and
  - if RHS(ρ) does not contain x_i, then $\tilde{q}_i = \bot$.
- for each $r = q(\#) \to u \in \delta_a$, $(\rho, \bar{q}) \to \# \in \delta'$ where $\bar{q} \to \# \in \delta^{\rho l}$.
- for each $\bar{q} \to \sigma(\bar{q}_1, \bar{q}_2) \in \delta^{rl}$,
  $(\bot, \bar{q}) \to \sigma((\bot, \bar{q}_1), (\bot, \bar{q}_2)) \in \delta'$.
- for each $\bar{q} \to \# \in \delta^{rl}$, $(\bot, \bar{q}) \to \# \in \delta'$.

- $A_s = (\delta_s \cup \{\bot\} \times Q^{rl}, \Sigma, \delta_{s0} \times Q^{rl}, \delta')$ can be constructed from T_s in the same way as A_a.

Step 3. Let Q_I be the set of states of A_I. Compute two subsets $\mathcal{Q}_a^D$ and $\mathcal{Q}_s^D$ of Q_I as follows:

It is also known that LDTT^R preserves regularity [7]. In the same way as Section 3.5.3, we can show that the revised algorithm for LDTT^{R}s works correctly. The sizes of A_a and A_s are exponential in $|T_a|$ and $|T_s|$, respectively, because $|A^{rl}|$ is exponential in $|\mathcal{A}|$ and $|A|$ is proportional to the product of $|A^{rl}|$, $|Q|$, and $|\delta|$. Thus, the time complexity of the algorithm is doubly exponential.

Theorem 3.5.8. *∞-SS is decidable in 2-EXPTIME for LDTT^R.*

3.6. Discussion

3.6.1 Multiple Authorized Queries

We have discussed ∞-SS for a single authorized query. Generalizing the results to the case that there are multiple authorized queries is important because using multiple authorized queries attackers can narrow down the candidates of secret information to less than using them independently in general. However, ∞-SS for more than one authorized queries is much complicated and might be undecidable at worst. In fact, the approach in Section 3.5.1 does not work well. Now we assume that LDTTs T_{a1} and T_{a2} are given as authorized queries, and T_s and A_G are given as an unauthorized query and a schema. Then, the following procedure which is almost the same as Section 3.5.2 is likely to work:

Step 1. Construct A_{a1}, A_{a2} and A_s from T_{a1}, T_{a2}, and T_s, respectively.

Step 2. Compute the product TA of A_{a1}, A_{a2}, A_s and A_G.

Step 3. Compute $\mathcal{Q}_{a1}^D$, $\mathcal{Q}_{a2}^D$ and $\mathcal{Q}_s^D$, respectively.

Step 4. Compute $Q' = \{q_I \in Q_I \mid q_I$ is recursive through a nonempty path such that all the states in the path are in $\mathcal{Q}_{a1}^D \cap \mathcal{Q}_{a2}^D$ and some state in the path is not in $\mathcal{Q}_s^D$.$\}$

Steps 5–6. These steps are the same as Section 3.5.2.

Step 7. Let $(T_{a1}, T_{a2})(L(A_{fin})) = \{(T_{a1}(t), T_{a2}(t)) \mid t \in L(A_{fin})\}$ and $(T_{a1}, T_{a2})(L(A_{inf})) = \{(T_{a1}(t), T_{a2}(t)) \mid t \in L(A_{fin})\}$. Then, decide whether $(T_{a1}, T_{a2})(L(A_{fin})) \subseteq (T_{a1}, T_{a2})(L(A_{inf}))$.

Unfortunately, the above procedure does not work in Step 7 because we do not know whether $(T_{a1}, T_{a2})(L(A_{fin})) \subseteq (T_{a1}, T_{a2})(L(A_{inf}))$ is decidable. At least, $(T_{a1}, T_{a2})(L(A_{inf}))$ cannot be captured by any regular tree language in general.

Similarly, even for a single authorized query, when the authorized query T_a is given by a *non-linear* top-down tree transducer, which allows to copy subtrees twice or more, $T_a(L)$ is not guaranteed to be regular for a regular tree language L. Thus, our algorithm cannot be extended directly to such cases.

3.6.2 k-Secrecy Preservation through Updates

We would like to extend k-SS for deciding the k-secrecy preservation through updates. For some schema, k-secrecy is too strong to satisfy. On the other hand, a database instance is frequently updated in the real world. Consider the property that for any instance t which is k-secret, every instance obtained from t through updates is also k-secret. We call this property k-secrecy preserving property through updates, abbreviated as k-SP. Formally, given an XML schema A_G, an authorized query T_a, an unauthorized query T_s, and update operations U, we say that U preserves k-secrecy with respect to (A_G, T_a, T_s) if $U(t)$ is k-secret whenever $t \in L(A_G)$ is k-secret with respect to (A_G, T_a, T_s). If a given schema, queries and update operations satisfy k-SP, then it suffices to guarantee that the initial instance is k-secret in order to keep every instance through updates k-secret.

We can show that a variant of ∞-SP is decidable by using the decidability result of ∞-SS. Consider a regular subset L_{sub} of $L(A_G)$ such that every instance in L_{sub} is ∞-secret. Here, for update operations, we adopt $\mathsf{UFO}_{\mathsf{reg}}$ [22], which is a restricted model for XML update primitives of the W3C XQuery Update Facility. The variant of ∞-SP problem is to decide whether all possible instances obtained by updating instances in L_{sub} by applying any finite number of update operations are k-secret or not. Since it is known that $\mathsf{UFO}_{\mathsf{reg}}$ has the forward regularity preserving property, the set $post^*(L_{sub})$ of the possible instances after

updates is also regular. Thus, to decide the variant of ∞-SP, we just decide whether $(post^*(L_{sub}) \cap L(A_G), T_a, T_s)$ is k-secret or not.

3.7. Conclusion

In this chapter, we have discussed the problem of deciding whether every database instance conforming to a given XML schema is safe against inference attacks for given authorized and unauthorized queries in the sense of k-secrecy (k-SS).

We have shown that k-SS is undecidable for any natural number $k > 1$ by reduction from PCP even when a query is given by a deterministic rational transducer on words (DRTW). For a positive result, we have shown that ∞-SS is EXPTIME-complete for LDTT and in 2-EXPTIME for LDTT^R. The proposed method can be applied to linear deterministic bottom-up tree transducer and a similar result has been obtained in [17].

Chapter 4

Query-Based ℓ-Diversity on Relational Databases

This chapter introduces a new privacy notion against inference attacks called *query-based ℓ-diversity* by extending ℓ-diversity [15] in relational databases with access control for queries. The basic notions of relational database is provided. Then we end with the experimental results conducted on two verification tools implemented by SQL and #SAT solver.

4.1. Introduction

As described in Chapter 2, there are well-known related notions such as k-anonymity and ℓ-diversity. Intuitively, a database is k-anonymous if for every individual x, there are at least k different records (or tuples in the relational database setting) which cannot be distinguished from the real record for x. A database is ℓ-diverse if for every individual x, there are at least ℓ different values of the sensitive information contained in the records which cannot be distinguished from the real record for x. Also, various methods are reported for transforming a given database into a database satisfying k-anonymity or ℓ-diversity. However, these notions do not take the effect of access control for queries into consideration.

The goal of this study is to introduce a notion of the security against inference attack by extending ℓ-diversity in relational databases with access control for queries. More specifically, we propose a new privacy notion called *query-based ℓ-*

diversity. A database instance T is (query-based) ℓ-diverse with respect to given authorized queries if an attacker cannot narrow down the number of possible values of the sensitive information for any individual to less than ℓ by inference based on the result of the authorized queries on the instance T and the queries themselves. We provide two approaches to deciding the query-based ℓ-diversity. In the first approach, a decision algorithm is given by using relational operations, which can be directly implemented by a relational database management system, e.g., SQL. The second approach transforms a given input to a logical formula and decides the problem by model counting using a #SAT solver. We discuss the effectiveness and scalability of the two approaches based on the experimental results.

4.2. Models

In this section, we introduce a simple relational database model , which will be used in the rest of this chapter. A relational database instance (or simply a database) can be seen as a *table*, of which columns are *attributes*. There are two types of attributes, namely *sensitive* and *nonsensitive* attributes. The values of sensitive attributes are considered as secret, that is, the data owner keeps them confidentially and restrictively and protects them from unauthorized accesses.

Definition 4.2.1. A *relational database schema* (or simply a schema) is a finite set of attributes. Let $\mathbf{R} = \{A_1, \ldots, A_n\}$ be a schema. We assume that for each attribute A_i $(1 \leq i \leq n)$, a finite set of values, denoted by $dom(A_i)$ is associated. A *tuple* (or a record) over $\mathbf{R}$ is $t = (d_1, \ldots, d_n)$ where $d_i \in dom(A_i)$ for each $1 \leq i \leq n$. Let $t[A_i] = d_i$, which is called the value of attribute A_i in t. That is, $t = (t[A_1], \ldots, t[A_n])$. A *relational database instance* (or simply an instance) of $\mathbf{R}$ is a finite set of tuples over $\mathbf{R}$. An instance is sometimes called a *table*. Let $I[\mathbf{R}]$ denote the set of all instances of $\mathbf{R}$.

Let $\mathbf{R}$ be a schema. We assume that $\mathbf{R}$ is divided into two disjoint subsets, namely, $\mathcal{S}e$ and $\mathcal{NS}e$, which are the set of *sensitive attributes* and the set of *nonsensitive attributes*, respectively. We furthermore assume that a subset $Qi \subseteq \mathcal{NS}e$ of nonsensitive attributes is given as a *quasi-identifier* of $\mathbf{R}$. Note that

the domain of every attribute is finite. We intend that the values of the quasi-identifier can be potentially used to identify the values of the sensitive attributes by linking the attribute values of the quasi-identifier with external data sets.

We define projection, selection and join in the usual way. Let $\mathbf{R}$ be a schema. For a subset of attributes $\alpha = \{A_{j_1}, \ldots, A_{j_m}\} \subseteq \mathbf{R}$ and a tuple t over $\mathbf{R}$, let $\pi_\alpha(t)$ denote the tuple $(t[A_{j_1}], \ldots, t[A_{j_m}])$, which is called the projection of t on α. Also, for an instance $T \in I[\mathbf{R}]$, let $\pi_\alpha(T) = \{\pi_\alpha(t) \mid t \in T\}$. Let $T_1 \in I(\mathbf{R}_1)$ and $T_2 \in I(\mathbf{R}_2)$. For a filtering condition F, and an instance T, let $\sigma_F(T)$ denote the set of tuples in T that satisfy F. The natural join of T_1 and T_2 is the instance obtained by "linking" every possible pair of tuples in T_1 and T_2:

$$T_1 \bowtie T_2 = \{t \textit{ over } \mathbf{R}_1 \cup \mathbf{R}_2 \mid \textit{for some } u \in T_1, \\ w \in T_2,\ t[\mathbf{R}_1] = u \textit{ and } t[\mathbf{R}_2] = w\}. \tag{4.1}$$

Since the natural join operator is associative and commutative, we sometimes view the natural join as a polyadic operator and write $T_1 \bowtie .... \bowtie T_m$.

4.3. Proposed Framework

In order to provide the definitions, we need to introduce the candidate set of instances of which results of queries are the same as those of the real instance. For a given instance T and queries $q_1, ..., q_m$, let $cand(q_1, ..., q_m, T)$ be the set consisting of all instances that give the same result as T with respect to all queries $q_1, ..., q_m$:

$$cand(q_1, ..., q_m, T) = \{T' \in I(\mathbf{R}) \mid \forall i(1 \leq i \leq m) \cdot \\ q_i(T) = q_i(T')\}. \tag{4.2}$$

Each $T' \in cand(q_1, ..., q_m, T)$ is called a candidate instance.

Let T be a database instance over schema $\mathbf{R}$. T is query-based ℓ-diverse if for each maximal subset of a candidate instance of which tuples have the same quasi-identifier, there are ℓ or more different values of the sensitive attributes.

Suppose that the following information is available to public: a database schema $\mathbf{R}$, authorized queries $q_1, ..., q_m$, quasi-identifier Qi, sensitive attributes

Se and a threshold ℓ (a positive integer). Let T be an instance of $\mathbf{R}$. An attacker infers sensitive information by taking the natural join of the results of the authorized queries $q_1, ..., q_m$ on the instance T to obtain the candidate set of sensitive information. We now show three options for the definition of query-based ℓ-diversity as follows.

Definition 4.3.1. An instance $T \in I(\mathbf{R})$ is (query-based) ℓ-diverse (with respect to $\mathbf{R}, Qi, Se, q_1, ..., q_m$)

(Option 1) if for every $t \in \pi_{Qi}(T)$,

$$|\{\pi_{Se}(t') \mid {}^{\exists}T' \in cand(q_1, ..., q_m, T) \cdot (\pi_{Qi}(t') = t \wedge t' \in T')\}| \geq l, \tag{4.3}$$

(Option 2) if for every $t \in \pi_{Qi}(T)$, there is an instance $T' \in cand(q_1, ..., q_m, T)$ such that

$$|\{\pi_{Se}(t') \mid (\pi_{Qi}(t') = t \wedge t' \in T')\}| \geq l, \tag{4.4}$$

(Option 3) if there is $T' \in cand(q_1, ..., q_m, T)$ such that for every $t \in \pi_{Qi}(T)$,

$$|\{\pi_{Se}(t') \mid (\pi_{Qi}(t') = t \wedge t' \in T')\}| \geq l. \tag{4.5}$$

By definition, (4.5) implies (4.4), and (4.4) implies (4.3).

A conjunctive query consists of projection, selection and join. In our proposed framework, we assume self-join free conjunctive queries.

Definition 4.3.2. A query q on $\mathbf{R}$ is *monotonic* if for any $T_1, T_2 \in I(\mathbf{R})$, $T_1 \subseteq T_2$ implies $q(T_1) \subseteq q(T_2)$.

Lemma 4.3.3. *Every conjunctive query is monotonic.*

If we restrict the class of queries to self-join free conjunctive queries, all the three definitions of ℓ-diversity become equivalent as stated in the next theorem.

Theorem 4.3.4. *If we assume self-join free conjunctive queries, then the three options in definitions 4.3.1 become equivalent.*

Proof. By the following properties 1 and 2. □

Property 1. For any instances T_1, T_2 and a self-join free conjunctive query q,

$$q(T_1 \cup T_2) = q(T_1) \cup q(T_2).$$

Proof. Let T_1, T_2 be instances and q be a self-join free conjunctive query. By Lemma 4.3.3, q is monotonic and hence $q(T_1 \cup T_2) \supseteq q(T_1) \cup q(T_2)$ holds. Since q does not contain self-join, $q(T_1 \cup T_2) \subseteq q(T_1) \cup q(T_2)$ also holds. □

Property 2. Let T be an instance and $q_1, \ldots, q_m$ be self-join free conjunctive queries. The largest candidate set in $cand(q_1, ..., q_m, T)$ (with respect to set inclusion) is the union of all instances in $cand(q_1, ..., q_m, T)$.

Proof. Let $T_c = \bigcup_{T' \in cand(q_1,...,q_m,T)} T'$. By Property 1,

$$\begin{aligned} q_i(T_c) &= \bigcup_{T' \in cand(q_1,...,q_m,T)} q_i(T') \\ &= \bigcup_{T' \in cand(q_1,...,q_m,T)} q_i(T) \\ &= q_i(T)\ (1 \leq i \leq m). \end{aligned}$$

Hence, $T_c \in cand(q_1, ..., q_m, T)$. Apparently, T_c is the largest set in $cand(q_1, ..., q_m, T)$. □

We define the query-based ℓ-diversity problem as follows:

Input : A schema $\mathbf{R}$, an instance $T \in I(\mathbf{R})$, authorized queries $q_1, ..., q_m$, quasi-identifier $Qi \subseteq \mathbf{R}$, sensitive attributes $Se \subseteq \mathbf{R}$, and a threshold $\ell \geq 1$.

Output : T is query-based ℓ-diverse or not with respect to $\mathbf{R}$, Qi, Se, $q_1, ..., q_m$.

4.4. Verification by Relational Algebra

In this section, we describe our verification algorithm that solves the query-based ℓ-diversity problem. For simplicity, we only focus on projection queries. The algorithm can be extended to deal with join without self-join. However, selection

cannot be allowed. Also, we assume that the set of sensitive attributes is not empty.

We assume that an attacker knows the domain of each attribute in $\mathbf{R}$, specially the domains of the sensitive attributes, so that he can infer a candidate instance by adding values of the sensitive attributes chosen from the domain even if (some of) the sensitive attributes are missing in the result of queries $q_1, ..., q_m$.

Our algorithm consists of four steps as follows:

1. Obtain the candidate set of tuples T' by taking the natural join of all results $q_1(T), ..., q_m(T)$ as follows.

$$T' = q_1(T) \bowtie ... \bowtie q_m(T).$$

2. Let Qi' ($\subseteq Qi$) be the set of quasi-identifier that exist in T'. Compute the subset T_c of T' consisting of tuples whose quasi-identifier value belongs to the original instance T.

$$T_c = T' \bowtie \pi_{Qi'}(T).$$

3. Divide T_c into subsets (equivalence classes) $g_1, \ldots, g_h$ such that

   (a) $\pi_{Qi'}(t) = \pi_{Qi'}(t')$ for any $t, t' \in g_i$ $(1 \leq i \leq h)$ and

   (b) $\pi_{Qi'}(t) \neq \pi_{Qi'}(t')$ for any $t \in g_i$ $(1 \leq i \leq h)$ and $t' \in g_j$ $(1 \leq j \leq h)$ with $i \neq j$.

4. Let mis_Se be the set of sensitive attributes that does not exist in T'. With mis_Se and the threshold ℓ, decide whether T is ℓ-diverse by examining the following necessary and sufficient condition for ℓ-diversity:
   $\forall g_i$ $(1 \leq i \leq h)$,

$$|g_i| \ \times \prod_{a \in mis\_Se} | \, dom(a) \, | \ \geq \ \ell. \tag{4.6}$$

In the last step, the number of different sensitive values in each equivalence class g_i $(1 \leq i \leq h)$ is computed by using the domains of the missing sensitive attributes mis_Se. If for every equivalence class g_i $(1 \leq i \leq h)$, the left-hand side

of (4.6) is greater than or equal to the threshold ℓ, the algorithm answers that the given input is ℓ-diverse. If there is at least one equivalence class g_i such that the left-hand side of (4.6) is less than ℓ, the algorithm answers that the given input is not ℓ-diverse. For a given database instance with n tuples and authorized queries $q_1, ..., q_m$, this decision algorithm takes $O(n^m)$ time.

4.5. Verification by Model Counting

In this section, we provide another method for deciding the query-based ℓ-diversity. The method transforms a given input of the problem to a logical formula presented in CNF and decides the problem by model counting using a #SAT solver to count the different values of sensitive attributes in the equivalence classes. This counting procedure terminates when it detects that the number of different values of sensitive attributes of an equivalence class is less than ℓ. The advantage of this method is that it can handle self-join free conjunctive queries, consisting of projection, selection and join without self-join. Henceforth, we assume queries in the class.

Before we explain our method, we give some definitions. For a formula Ψ, let $\#models(\Psi)$ denote the number of different models (assignments to variables that make Ψ true). If a formula Ψ contains only variables in Σ, we call Ψ a Σ-formula. For a Σ-formula Ψ and $\Delta \subseteq \Sigma$, let $\Psi_{|\Delta}$ denote the strongest Δ-formula implied by Ψ when considered as a Σ-formula where A is stronger than B if and only if $A \Rightarrow B$ holds. We say that $\Psi_{|\Delta}$ is the projection of Ψ onto Δ.

Assume that a schema $\mathbf{R}$ where $n = |\mathbf{R}|$, an instance $T \in I(\mathbf{R})$, queries $q_1, \ldots, q_m$ on $\mathbf{R}$, quasi-identifies $Qi \subseteq \mathbf{R}$, sensitive attributes $Se \subseteq \mathbf{R}$, and a threshold ℓ are given. For simplicity, suppose that $Qi = \{A_1, \ldots, A_k\} \subseteq \mathbf{R}$, and $Se = \{A_{k+1}, \ldots, A_m\} \subseteq \mathbf{R}$ where $1 \leq k < m < n$. The summary of the method is as follows.

1. Construct a logical formula $\Phi(x_1, \ldots, x_n)$ such that

   $\Phi(c_1, \ldots, c_n)$ is satisfiable
   if and only if $(c_1, \ldots, c_n) \in T_c \quad (*)$

   Note that $\phi(x_1, \ldots, x_n)$ has free variables other than $x_1, \ldots, x_n$ in general.

2. Decide if for all tuple $(c_1, \ldots, c_k) \in \pi_{Qi}(T)$,

$$\#models(\Phi_p(x_{k+1}, \ldots, x_n)_{|Xs}) \geq l.$$

where $Xs = \{x_{k+1}, \ldots, x_m\}$ and

$$\Phi_p(x_{k+1}, \ldots, x_n) = \Phi(c_1, \ldots, c_k, x_{k+1}, \ldots, x_n).$$

Constructing Constraint

Let $n = |\mathbf{R}|$ and $n_i = |\mathbf{R_i}|$ where $\mathbf{R_i}$ is the output schema of q_i $(1 \leq i \leq m)$. To construct a formula $\Phi(x_1, \ldots, x_n)$ satisfying $(*)$, we first construct subformulas ϕ_{q_i} and O_{q_i} for $1 \leq i \leq m$.

(1-i) For $1 \leq i \leq m$, ϕ_{q_i} represents the input-output relation of the query q_i. The formula ϕ_{q_i} contains free variables $x_1, \ldots, x_n, y_1, \ldots, y_{n_i}$ and satisfies:

for any $t = (c_1, \ldots, c_n)$ and $t'_i = (d_1, \ldots, d_{n_i})$,
$\phi_{q_i}(c_1, \ldots, c_n, d_1, \ldots, d_{n_i})$ is satisfiable if and only if $q_i(\{t\}) \subseteq \{t'\}$.

(Construction)
If $q = T$ then

$$\phi_q(x_1, \ldots, x_n, y_1, \ldots, y_n) = \bigwedge_{i=1}^{n}(x_i = y_i).$$

projection: If $q = \pi_\alpha(q')$ where $\alpha = \{A_{j_1}, \ldots, A_{j_{n'}}\}$,

$$\begin{aligned}&\phi_q(x_1, \ldots, x_{n_I}, z_1, \ldots, z_{n'}) \\ &= \phi_{q'}(x_1, \ldots, x_{n_I}, y_1, \ldots, y_{n_O}) \wedge \bigwedge_{i=1}^{n'}(y_{j_i} = z_i).\end{aligned}$$

selection: If $q = \sigma_F(q')$,

$$\begin{aligned}&\phi_q(x_1, \ldots, x_{n_I}, z_1, \ldots, z_{n_O}) \\ &= \phi_{q'}(x_1, \ldots, x_{n_I}, y_1, \ldots, y_{n_O}) \\ &\quad \wedge \left(P_F(y_1, \ldots, y_{n_O}) \Rightarrow \bigwedge_{i=1}^{n_O}(y_i = z_i)\right).\end{aligned}$$

where $P_F(y_1, \ldots, y_{n_O})$ is a formula representing the filtering condition F of σ_F.

cross product: If $q = q' * q''$,

$$\begin{aligned}
&\phi_q(x_1, \ldots, x_{n'_I}, x'_1, \ldots, x'_{n''_I}, z_1, \ldots, z_{n'_O+n''_O}) \\
&= \phi_{q'}(x_1, \ldots, x_{n'_I}, y_1, \ldots, y_{n'_O}) \\
&\quad \wedge \phi_{q''}(x'_1, \ldots, x'_{n'_I}, y'_1, \ldots, y'_{n''_O}) \\
&\quad \wedge \bigwedge_{i=1}^{n'_O}(y_i = z_i) \wedge \bigwedge_{i=1}^{n''_O}(y'_i = z_{n'_O+i}).
\end{aligned}$$

(1-ii) O_{q_i} is defined as

$$\begin{aligned}
&O_{q_i}(y_1, \ldots, y_{n_i}) \\
&= \bigvee_{(d_1,\ldots,d_{n_i}) \in q_i(T)} ((y_1 = d_1) \wedge \cdots \wedge (y_{n_i} = d_{n_i})).
\end{aligned}$$

(1-iii) Finally, Φ is defined as

$$\begin{aligned}
&\Phi(x_1, \ldots, x_n) \\
&= \bigwedge_{i=1}^{m}(\phi_{q_i}(x_1, \ldots, x_n, y_{i,1}, \ldots, y_{i,n_i}) \wedge O_{q_i}(y_{i,1}, \ldots, y_{i,n_i})).
\end{aligned}$$

Remember that in the algorithm of the previous section, we introduce the subsets $g_1, \ldots, g_h$, each of which shares same values of the quasi-identifier. For g_j $(1 \leq j \leq h)$, let $(c^j_1, \ldots, c^j_k)$ be the values of the quasi-identifier shared by tuples in g_j. Let $\Phi^j_p(x_{k+1}, \ldots, x_n) = \Phi(c^j_1, \ldots, c^j_k, x_{k+1}, \ldots, x_n)$. By $(*)$, $\Phi^j_p(c_{k+1}, \ldots, c_n)$ is satisfiable if and only if $(c^j_1, \ldots, c^j_k, c_{k+1}, \ldots, c_n) \in g_j$. Furthermore, $\Phi^j_p(x_{k+1}, \ldots, x_n)_{|Xs}$ is the strongest Xs-formula implied by $\Phi^j_p(x_{k+1}, \ldots, x_n)$. Hence, the number of assignments to variables in Xs that make $\Phi^j_p(x_{k+1}, \ldots, x_n)_{|Xs}$ true coincides with the number of different values of Se appearing in tuples that belong to g_j. Hence, we obtain the following lemma.

Lemma 4.5.1. *Let* $\mathbf{R}$ *be a schema,* $Qi, Se \subseteq \mathbf{R}$ *be the quasi-quantifier and sensitive attributes, respectively,* $q_1, \ldots, q_m$ *be self-join free conjunctive queries on* $\mathbf{R}$ *and* $T \in I(\mathbf{R})$ *be an instance. Let* $g_1, \ldots, g_h$ *be the subsets of* T_c*, each of which shares same values for the quasi-identifier. For each* j $(1 \leq j \leq h)$*, the number of different values of sensitive attributes in* g_j *is*

$$\#models(\Phi^j_p(x_{k+1}, \ldots, x_n)_{|Xs}).$$

Counting Candidates

To count the different values of sensitive attributes for each g_j $(1 \leq j \leq h)$, we transform $\Phi_p^j(x_{k+1}, \ldots, x_n)$ to an equivalent propositional formula Φ_{cnf}^j in conjunctive normal form (CNF) by using Sugar [20]. Next, for each $t' = (c_1, \ldots, c_k) \in \pi_{Qi}(T)$, we construct a CNF formula $\psi_{t'}$ that represents $x_1 = c_1 \wedge \cdots \wedge x_k = c_k$, and then count $\#models(\Phi_{cnf}^j \wedge \psi_{t'})_{|P(Xs)}$, where $P(Xs)$ is the set of the propositional variables in Φ_{cnf}^j corresponding to Xs in $\Phi_p^j(x_{k+1}, \ldots, x_n)$. We use sharpCDCL [13], which is a #SAT solver (an automatic tool for counting the models of a given propositional formula). Among other #SAT solvers that can count models, the advantage of sharpCDCL is that it can automatically count $\#models(\Psi_{|\Delta})$ only by giving a formula Ψ and a subset Δ of propositional variables. If some $t' \in \pi_{Qi}(T)$ such that $\#models(\Phi_{cnf}^j \wedge \psi_{t'})_{|P(Xs)} < \ell$ is found, we say that T is not ℓ-diverse. Otherwise, T is ℓ-diverse. For a given propositional formula Φ_{cnf}^j over n variables, the model counting algorithm counts the distinct truth assignments to variables that make Φ_{cnf}^j true, and hence this counting problem is solvable in #P.

4.6. Experiments

The purpose of the experiment was to investigate the scalability and effectiveness of the algorithms for deciding ℓ-diversity problem.

4.6.1 Experimental Result of Relational Algebra

Setup

Experiment were performed on a 3.33 GHz Intel(R) Core(TM) i7 CPU with 6GB of RAM. The operating system was Microsoft Windows 8.1 Enterprise, and implementation was built and run in MySQL Workbench, version 6.1. We used available dataset, Employees Sample Database [23], Copyright (C) 2007, 2008, MySQL AB, version 1.0.6. The database contains about 300,000 tuples with 2.8 million salary entries. In our experiment, the schema consists of ten attributes, where five attributes $\{Gender,\ DeptName,\ BirthDate,\ HireDate,\ FromDate\}$ were designated as the quasi-identifier and the sensitive attribute is $\{Salary\}$.

Datasets and Queries

The proposed algorithm was implemented in MySQL and was performed on three instances (datasets) with $n = 37,500, 75,000, 150,000, 300,000$ tuples. Also, we prepared three queries, each of which is the projection onto the following attributes:

$q_1 : \{EmpNo, LastName, Gender\}$.
$q_2 : \{EmpNo, Salary, HireDate\}$.
$q_3 : \{DeptName\}$.

In the experiment, we used three sets of queries, namely, $\mathbf{Q_A} = \{q_1, q_2\}$, $\mathbf{Q_B} = \{q_3\}$, and $\mathbf{Q_C} = \{q_1, q_2, q_3\}$. For example, for $\mathbf{Q_A}$, the verification algorithm took the natural join of the results of q_1 and q_2 on each of the datasets in Step 1. In Step 3, the algorithm constructed the table from the candidate set T_c obtained in Step 2 by grouping tuples that have same values of the quasi-identifier. Lastly, in Step 4, the algorithm tested ℓ-diversity ($\ell = 2$ in the experiment).

Table 4.1. Total time of verifying 2-diversity.

Dataset	Cases	Total time
$37,500$	q_1, q_2	7sec
	q_3	4sec
	q_1, q_2, q_3	11sec
$75,000$	q_1, q_2	19sec
	q_3	8sec
	q_1, q_2, q_3	31sec
$150,000$	q_1, q_2	1min 7sec
	q_3	14sec
	q_1, q_2, q_3	1min 47sec
$300,000$	q_1, q_2	4min 8sec
	q_3	26sec
	q_1, q_2, q_3	8min 48sec

Table 4.1 shows the total running time of our algorithm. For example, for $\mathbf{Q_A}$ and the dataset $n = 37,500$, the total running time is 7sec. Also the

running time of each set of queries, $\mathbf{Q_A}, \mathbf{Q_B}$ and $\mathbf{Q_C}$ on the datasets of size $n = 37,500, 75,000, 150,000$ and $300,000$. We can observe that the decision algorithm is efficient, in general, and also the computation time depends on the size of the datasets.

4.6.2 Experimental Result of Model Counting

Setup

Experiment were performed on a 3.10 GHz Intel(R) Core(TM) i5 CPU with 8GB of RAM. The operating system was Ubantu 14.04. We performed the experiment on a dataset, having 50,000 tuples.

Datasets and Queries

In the experiment, we used two instances T_1 and T_2, having ten and eleven attributes, respectively. Both of T_1 and T_2 have $5,000$ tuples. Actually, T_1 was obtained from T_2 by projecting out one of the eleven attributes. We conducted the experiment on the following two settings:

$D1$: a query $\sigma_{state=Iwate}(T_1)$,
$Qi = \{ID, state\}$ and $Se = \{Name\}$.

$D2$: two queries $\Pi_{BirthYear,BirthMonth}(T_2)$,
$\Pi_{BirthYear,BirthMonth}(\sigma_{Carrier=SoftBank}(T_2))$,
$Qi = \{ID, BirthMonth, Carrier\}$
and $Se = \{BirthYear\}$.

The experimental results for these settings are shown in Table 4.2 where clauses and variables are those in the transformed CNF formula, projected variables are the variables corresponding to sensitive attributes, min count is the minimum number of different values of sensitive attributes among $g_1, \ldots, g_h$. That is, D1 is ℓ-diverse if and only if $\ell \leq 50$ and D2 is ℓ-diverse if and only if $\ell \leq 42$. Next, we increased the number of tuples in T_2 to $10,000$, $30,000$ and $50,000$ and examined the scalability of the proposed method by using the setting $D2$. The result is shown in Table 4.3. In sharpCDCL, an upperbound U of the model counting can be specified. That is, when sharpCDCL detects that the current number of models reaches U, sharpCDCL terminates. The computation times in Table 4.3

Table 4.2. Performance of model counting method.

	Clauses	Variables	Projected variables	Min count	Time
$D1$	$34,271$	$14,916$	$4,961$	50	6min 15sec
$D2$	$22,941$	$11,121$	96	42	2min 1sec

are those when this upperbound is specified as $U = 20$. The transformation to

Table 4.3. Scalability of model counting method.

Tuples	Clauses	Variables	Projected variables	Time
$5,000$	$22,941$	$11,121$	96	1min 58sec
$10,000$	$42,334$	$20,877$	96	10min 2sec
$30,000$	$117,557$	$58,541$	106	2hrs 28min 30sec
$50,000$	$187,820$	$93,633$	106	8hrs 56min 8sec

a CNF formula takes less than one second, and model counting dominates the running time.

4.6.3 Additional Result

We compare the results of two approaches by observing the total running time of the smallest dataset and queries in Table 4.5. Additionally, the total running time of the first approach on datasets is shown in Table 4.4.

Setup

The experiment were run on a different environment, as mentioned in Section 4.6.2 and 4.6.1, independently.

Datasets and Queries

The experiment performed on the same datasets as in Section4.6.1 and queries sets were given as follows, where $\mathbf{Q_A} = \{q_4\}$, $\mathbf{Q_B} = \{q_5, q_6\}$, and $\mathbf{Q_C} = \{q_6, q_7, q_8\}$.

$q_4 : \{HireDate, LastName, Gender, Salary\}$.
$q_5 : \{HireDate, LastName, Gender\}$.
$q_6 : \{HireDate, Salary\}$.
$q_7 : \{HireDate, LastName\}$.
$q_8 : \{LastName, Gender\}$.

Table 4.4. Total running time of relational algebra.

Dataset	Cases	Total time
$37,500$	q_4	7sec
	q_5, q_6	19sec
	q_6, q_7, q_8	32sec
$75,000$	q_4	10sec
	q_5, q_6	50sec
	q_6, q_7, q_8	1min 33sec
$150,000$	q_4	1min 16sec
	q_5, q_6	2min 54sec
	q_6, q_7, q_8	6min
$300,000$	q_4	4min 28sec
	q_5, q_6	12min 47sec
	q_6, q_7, q_8	22min 05sec

Table 4.5. Computation time of two approaches on a dataset.

Dataset	Cases	Relational algebra	Model counting
$37,500$	q_4	7sec	3min 32sec
	q_5, q_6	19sec	1min 36sec
	q_6, q_7, q_8	32sec	38sec

Table 4.5 confirms that verification method for security based on relational algebra using SQL is faster than model counting using #SAT solver.

4.7. Conclusion

We have introduced query-based ℓ-diversity as a privacy notion for a realistic database system that assumes access control for queries. This new notion inherits from ℓ-diversity of [15] the quantitative notion for the diversity of sensitive attributes. Also, the notion utilizes k-secrecy of [11] by taking attacker's inference on the authorized information into consideration.

We proposed two approaches to deciding whether a given database instance satisfies query-based ℓ-diversity with respect to given queries. The first approach is based on relational algebra computation that counts the candidate values of the sensitive attributes. The second approach transforms a given input to a logical formula and then decide the problem by counting models of a formula by a #SAT solver. The first approach can directly be implemented by an existing relational database system such as SQL, and the experimental results show that this approach is fairly efficient. The weakness is that it cannot deal with selection queries. The second approach, on the other hand, can deal with selection queries. However, the model counting in a #SAT solver is generally time consuming and we have not yet customized the solver to our problem and hence, the performance is not good compared with the first approach.

Chapter 5

Conclusion

Security against inference attacks has been the main motivation in this dissertation. When releasing data in a database system, it is necessary to prevent the sensitive information of an individual from being disclosed either directly or indirectly. Access control can perfectly control the direct disclosure of the data, but it cannot avoid the indirect disclosure in general. An inference attack is a typical way of indirectly leaking the sensitive information. For defining the security against inference attacks, we need a quantitative notion for security and privacy because the perfect security or privacy cannot be achieved when indirect leakage or inference exists. Quantitative notions such as k-anonymity and ℓ-diversity were proposed and widely used. However, these notions do not take access control into consideration. The motivation of this dissertation is to propose quantitative notions for security and privacy for a database system with access control.

More concretely, this dissertation has presented two studies on verifying the security against inference attacks on both XML and relational databases.

In Chapter 2, we reviewed quantitative notions for security and privacy, namely, k-anonymity, ℓ-diversity and k-secrecy. The first two are well-known notions mainly for relational databases, which are defined based on the number of tuples or the distribution of the sensitive information included in the tuples indistinguishable from one another. Although these notions are simple and useful, they cannot be directly used in databases having structural data or databases with access control. k-secrecy assumes access control in terms of authorized and unauthorized queries and is defined based on the number of candidates of the

results of the unauthorized query (the sensitive information) obtained by an inference attack. k-secrecy is a security notion that can be applied not only to relational databases, but also more general databases such as XML databases. The remaining chapters utilized and extended these notions to investigate verification methods for those extended notions as follows.

Chapter 3 discussed schema-level k-secrecy for XML databases. Based on instance-level k-secrecy introduced by Hashimoto et al., we first defined schema-level k-secrecy. Schema-level k-secrecy is important because once we can guarantee that a schema is k-secret, we do not have to check the k-secrecy of an instance conforming to the schema when an update of a database instance takes place. We showed that k-secrecy is undecidable for any finite k when queries are given by linear deterministic top-down tree transducers (LDTTs). Also we showed that ∞-secrecy is decidable in the same setting. A similar positive result of a stronger query class, linear top-down tree transducers with regular look-ahead (LDTT^R) was obtained. We also analyzed the time complexity of the decision problem, and showed that the problem is EXPTIME-complete.

Chapter 4 discussed query-based ℓ-diversity for relational databases. As mentioned above, k-anonymity and ℓ-diversity do not take access control into consideration. This chapter first introduced a new notion, query-based ℓ-diversity by incorporating the way of quantifying the security in defining k-secrecy into ℓ-diversity. A database instance is (query-based) ℓ-diverse with respect to given authorized queries if there are at least ℓ different values of sensitive information for an individual, which are obtained by inference based on the available query results. It is obvious that query-based ℓ-diversity is decidable for relational databases. In this dissertation, we proposed two approaches to deciding query-based ℓ-diversity, namely, an approach that directly uses relational algebra and an approach based on model counting of a propositional formula. The second approach can deal with self-join free positive conjunctive queries while the first approach cannot deal with queries including selection operations. The scalability and effectiveness of the two approaches were discussed based on the experimental results conducted on the implemented tools. The first approach was implemented by SQL and the second approach was implemented by #SAT solver. For the scalability in efficiency, the experimental results emphasized that the verification by

relational algebra was much faster than the other approach. As shown in the additional experimental results, the model counting approach for the dataset with $37,500$ tuples had a total running time nearly identical to the relational algebra approach on the dataset with $75,000$ tuples.

Chapter 3 assumed that the number of authorized queries is one for simplicity. It is left as a future study to extend our algorithm to work with multiple authorized queries and investigate the computational complexity of the extended problem. We also want to extend the notion of schema k-secrecy to k-secrecy preservation (k-SP) through updates. Investigating the decidability of k-SP in an appropriate model of update operations is interesting future work. Also, applying the approaches proposed in Chapter 4 to other kind of databases such as object-oriented or XML databases is challenging future work. Furthermore, the model counting approach was slower than relational algebra approach. In particular, the model counting approach took very much time when query results were large because the size of the query results directly affects the size of the translated CNF formula. To overcome this problem, we want to improve the translation algorithm so that a given input can be translated into a smaller CNF formula.

Acknowledgements

Studying in Japan had been my dream since I was a high school student. Six years ago, my dream came true when I received an opportunity to come to Japan as a graduate student at Nara Institute of Science and Technology. Achieving a doctoral degree would not have been possible without this opportunity and the support from many people; I could not have been able to complete this dissertation, if it were not for them.

First of all, I would like to express my sincere thanks to my supervisor, Professor Minoru Ito for his guidance and helpful comments for the improvement of this dissertation. And I sincerely appreciate that he kindly welcomed me to be a part of Ito-ken in my last year of doctoral course.

I would like to thank Professor Kenichi Matsumoto for his kind advice and useful comments for improving this dissertation, especially during my midterm and public defense presentation. His feedback is highly appreciated.

I would like to express my deep gratitude to Professor Hiroyuki Seki of Nagoya University, who was my previous supervisor, who, first and foremost, opened the doors to master and doctoral courses to me. He made it possible for me to come to Japan and pursue my degrees in his laboratory, Seki-ken. I feel very heartily thankful to him, who introduced me to this research field and showed me the importance of this research topic, who is always being patient in teaching and guiding me a lot and a lot, who reviews my publications and theses meeting after meeting. His invaluable advices, suggestions, and concerns have always motivated me and pushed me towards a goal with productive outcomes. I have been and always will be motivated to do work with him. I sincerely thank him for sharing his research visions and experiences with me.

I would also like to thank Associate Professor Yuichi Kaji for his invaluable

comments and kind support for improving this research, as well as his guidance of how to enjoy doing research and how to be more productive.

I would like to extend my deep gratitude to Assistant Professor Kenji Hashimoto of Nagoya University for his continuous teaching on both technical and non-technical issues for the past five years. This work could not have been accomplished without his well-directed advice, helpful suggestions, fruitful discussions, meetings, thesis review, and all the help with my publications; even though sometimes he had to explain things to me again and again, he never gave up on me. I have learned many valuable aspects of being a researcher from his guidance toward study and have always enjoyed doing research with him.

I am sincerely indebted to Japanese government, especially AYF and MEXT, for financially supporting my master and doctoral courses in NAIST.

I would like to thank members of Seki-ken and Ito-ken for being so supportive. To my AYF family, without your support, I would not have been able to speak Japanese and lived an easier life in Japan. To my Filipino family and friends in NAIST, without you I would not have learned to live my life in NAIST enjoyably and productively. To Friday Club, LOUIE, MARC, RYAN, LUPEE, MICHAEL, JASON, ERLYN, and BRYAN#todahappiness; you guys made my day funnier, crazier, and brighter! Thanks for being a family. To the International Student Affairs Section in NAIST, without you, I would get in trouble during my stay in NAIST. Especially to the Lao community in Japan, without you I would get homesick all the time.

Last but not least, I can never thank my family enough: my beloved father, mother, sister, and brother for their endless love, cares, and prayers for my success. I am forever indebted to them for making my life so meaningful and successful. Behind both my achievements and difficult times are their limitless support and understanding. I thank my relatives and my lovely friends in Laos for their unconditional support, especially when I needed it the most.

References

[1] R. J. Bayardo and R. Agrawal, "Data Privacy through Optimal k-Anonymization," 10th International Conference on Database Theory (ICDT), 2005, pp. 217-228.

[2] E. Bertino and R. Sandhu, "Database Security Concepts, Approaches, and Challenges," IEEE Transactions on Dependable and Secure Computing, vol. 2, no. 1, 2005, pp. 2-18.

[3] G. J. Bex, S. Maneth, and F. Neven, "A Formal Model for an Expressive Fragment of XSLT," Lecture Notes in Artificial Intelligence, vol. 1861, 2000, pp. 1137-1151.

[4] J. W. Byun, A. Kamra, E. Bertino, and N. Li, "Efficient k-Anonymization Using Clustering Techniques," 12th International Conference on Database Systems for Advanced Applications (DASFAA), 2007, pp. 188-200.

[5] H. Comon, M. Dauchet, R. Gilleron, F. Jacquemard, D. Lugiez, C. Löding, S. Tison, and M. Tommasi, "Tree Automata Techniques and Applications," 2008, URL: http://tata.gforge.inria.fr/.

[6] A. Deutsch and Y. Papakonstantinou, "Privacy in Database Publishing," 10th International Conference on Database Theory (ICDT), 2005, pp. 230-245.

[7] J. Engelfriet, "Top-Down Tree Transducers with Regular Look-Ahead," Theory of Computing Systems, vol.10, no.1, 1976, pp. 289-303.

[8] B. C. M. Fung, K. Wang, R. Chen, and P. S. Yu, "Privacy-Preserving Data Publishing: A Survey of Recent Developments," ACM Computing Surveys, vol. 42, 2010, pp. 14:1-53.

[9] H. Hosoya, "Foundations of XML Processing: The Tree-Automata Approach," Cambridge University Press, 2010.

[10] K. Hashimoto, H. Kawai, Y. Ishihara, and T. Fujiwara, "Decidability of the Security against Inference Attacks Using a Functional Dependency on XML Databases," IEICE Transactions on Information and Systems, vol. E95-D, no. 5, 2012, pp. 1365-1374.

[11] K. Hashimoto, K. Sakano, F. Takasuka, Y. Ishihara, and T. Fujiwara, "Verification of the Security Against Inference Attacks on XML Databases," IEICE Transactions on Information and Systems, vol. E92-D, no. 5, 2009, pp. 1022-1032.

[12] S. Kepser, "A Simple Proof for the Turing-Completeness of XSLT and XQuery," Extreme Markup Languages, 2004.

[13] V. Klebanov, N. Manthey, and C. Muise, "SAT-Based Analysis and Quantification of Information Flow in Programs," 10th International Conference on Quantitative Evaluation of Systems (QEST), 2013, pp. 177-192.

[14] N. Li, T. Li, and S. Venkatasubramanian, "t-Closeness: Privacy Beyond k-Anonymity and l-Diversity," 23rd IEEE International Conference Data Engineering (ICDE), 2007, pp. 106-115.

[15] A. Machanavajjhala, J. Gehrke, D. Kifer, and M. Venkitasubramaniam, "l-Diversity: Privacy Beyond k-Anonymity," 22nd International Conference on Data Engineering (ICDE), 2006, also in ACM Transactions on Knowledge Discovery from Data (TKDD), 2007, vol. 1(3), 52 pages.

[16] G. Miklau and D. Suciu, "A Formal Analysis of Information Disclosure in Data Exchange," Journal of Computer and System Sciences, vol. 73, 2007, pp. 507-534.

[17] C. Phonharath, "Verification of the Security against Inference Attacks on XML Databases," Master's Thesis, NAIST-IS-MT1051135, Nara Institute of Science and Technology, 2012,
http://library.naist.jp/library/thesis/is/ism2012.html.

[18] C. Phonharath, K. Hashimoto and H. Seki, "Deciding Schema k-Secrecy for XML Databases," IEICE Transactions on Information and Systems, vol. E96-D, 2013, pp. 1268-1277.

[19] P. Samarati, "Protecting Respondents' Identities in Microdata Release," IEEE Transactions on Knowledge and Data Engineering, vol. 13, 2001, pp. 1010-1027.

[20] "A SAT-Based Constraint Solver," URL: http://bach.istc.kobe-u.ac.jp/sugar/[accessed: 2015-01-23].

[21] L. Sweeney, "k-Anonymity: A Model for Protecting Privacy," International Journal on Uncertainty, Fuzziness and Knowledge-Based Systems, vol. 10, 2002, pp. 557-570.

[22] F. Jacquemard, M. Rusinowitch, "Rewrite-Based Verification of XML Updates," 12th International ACM SIGPLAN Symposium on Principles and Practice of Declarative Programming (PPDP), 2010, pp. 119-130.

[23] F. Wang and C. Zaniolo, "Employees Sample Database," 2008, URL: https://launchpad.net/test-db/[accessed: 2014-11-13].

[24] X. Xiao and Y. Tao, "Personalized Privacy Preservation," ACM SIGMOD international conference on Management of data (SIGMOD), 2006, pp. 229-240.

Publication List

Peer-reviewed Journal Paper

1. Chittaphone Phonharath, Kenji Hashimoto, and Hiroyuki Seki, "Deciding Schema k-Secrecy on XML Databases," IEICE Transactions on Information and Systems, vol. E96-D, no. 6, 2013, pp. 1268-1277. (related to Chapter 3)

Peer-reviewed International Conference Papers

1. Chittaphone Phonharath, Kenji Hashimoto, and Hiroyuki Seki, "Verification of the Security against Inference Attacks on XML Databases," 1st International Workshop on Trends in Tree Automata and Tree Transducers (TTATT), 2012, pp. 11-22. (related to Chapter 3)

2. Chittaphone Phonharath, Ryonosuke Takayama, Kenji Hashimoto, and Hiroyuki Seki, "Query-Based ℓ-Diversity," 7th International Conference on Advances in Databases, Knowledge, and Data Applications (DBKDA), 2015, pp. 15-20. (related to Chapter 4)

Workshop Papers

1. Chittaphone Phonharath, Kenji Hashimoto, and Hiroyuki Seki, "Static Analysis for k-Secrecy against Inference Attacks," IEICE Technical Report SS2011-4, vol. 111, no. 107, 2011, pp. 17-22. (related to Chapter 3)

2. Chittaphone Phonharath, Ryonosuke Takayama, Kenji Hashimoto, and

Hiroyuki Seki, "Query-Based ℓ-Diversity," IEICE Technical Report SS2015-14, vol. 115, no. 20, 2015, pp. 65-70. (related to Chapter 4)