Vol.2013-MPS-94 No.10
2013/7/22

Decidability of k-secrecy against inference attacks using functional dependencies on XML databases

Nobuaki Yamazoe[1] Kenji Hashimoto[2,a)] Yasunori Ishihara[1,b)] Toru Fujiwara[1,c)]

Abstract: An inference attack means that a database user tries to identify, or narrow down the candidates for, sensitive information from non-sensitive information such as queries authorized to the user and their results, the schema of a database, functional dependencies satisfied by the database, etc. If the size of the candidate set is at least k, the database is said to be k-secret. In our previous papers, we targeted XML databases and proposed how to determine k-secrecy without functional dependencies. In this paper, we show the decidability of k-secrecy with functional dependencies provided that the functional dependencies satisfy a restriction called the non-prefix restriction. To be specific, we reduce the problem of finding a candidate to the satisfiability problem of functional dependencies. Then, the decision algorithm of k-secrecy is simply designed as an enumeration of candidates.

1. Introduction

Direct access to a database is controlled in general. That is, database management systems specify which users can issue which queries. However, by using non-sensitive information such as authorized queries and their results, the schema of the database, and the functional dependencies satisfied by the database, a user may be able to identify, or narrow down the candidates for, the result of some unauthorized query. Such indirect access to the result of an unauthorized query is called an *inference attack*. In order to maintain the secrecy of the database, it is important for database managers to know of possible inference attacks in advance. Below is an example of an inference attack.

Example 1 Consider an XML database containing information on patients in a hospital. Suppose that three queries T_1, T_2, and T_3 are authorized to a user:

- T_1 retrieves all the patients examined by Dr. Abe and the day of the week of the examination;
- T_2 retrieves all the patients in room 101 and the day of the week of the examination; and
- T_3 retrieves all the doctors who examine a patient in room 102.

Also suppose that this XML database satisfies the following two functional dependencies f_1 and f_2:

- f_1: the day of the week of the examination uniquely determines the doctor; and
- f_2: the room uniquely determines the disease of the patients in the room.

The user is interested in the result of the following query T_S:

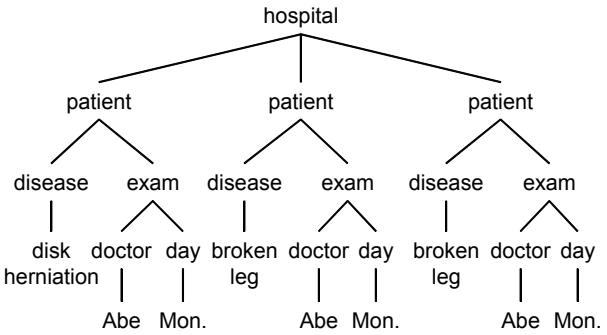

Fig. 1 The result of query T_1.

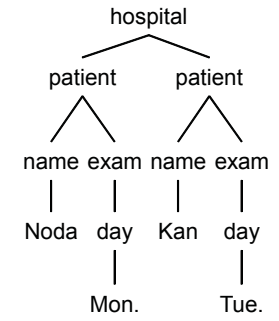

Fig. 2 The result of query T_2.

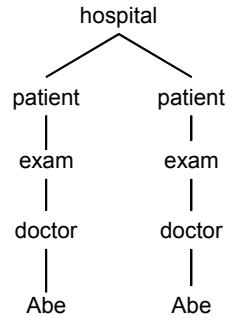

Fig. 3 The result of query T_3.

- T_S retrieves the disease of patient Noda.

Suppose that T_S is not authorized to the user, so the user attempts indirect access to the result of T_S.

Now, suppose that the results of T_1, T_2, and T_3 are the trees shown in Figures 1, 2, and 3. We can see in Figure 3 that there are two patients in room 102 examined by Dr. Abe. Moreover, by f_2, these two patients have the same disease. Figure 1 indicates that Dr. Abe examines three patients. Since two of the three patients are in room 102 and have the same disease, the patient with

[1] Osaka University, Suita, Osaka 565–0871, Japan
[2] Nara Institute of Science and Technology, Ikoma, Nara 630–0192, Japan
a) k-hasimt@is.naist.jp
b) ishihara@ist.osaka-u.ac.jp
c) fujiwara@ist.osaka-u.ac.jp

disk herniation is in a room other than 102. By f_1 and Figures 1 and 2, it can be concluded that Dr. Abe examines patient Noda. Moreover, since the patient with disk herniation is the only patient examined by Dr. Abe other than the patients in room 102, it can be inferred that patient Noda has disk herniation.

Note that in this example, if f_2 is unavailable to the user, the disease of patient Noda cannot be identified but the candidates are narrowed down to disk herniation and broken leg. On the other hand, if f_1 is unavailable to the user, it is impossible to even narrow down the candidates.

If the number of candidates narrowed down by the attacker is large, it is hard to identify which candidate is the true value and the database is considered safe. If the size of the candidate set is at least k, the database is *k-secret*, and if the size of the set is not finite, the database is *infinity-secret* [7]. Aiming at XML databases, we previously proposed how to determine infinity-secrecy and k-secrecy without functional dependencies [7]. We also proposed how to determine infinity-secrecy with a single functional dependency [6]. However, it remains an open problem whether k-secrecy with functional dependencies is decidable or not.

In this paper, we show the decidability of k-secrecy with multiple functional dependencies provided that the functional dependencies satisfy a restriction called the *non-prefix restriction*. Roughly speaking, the non-prefix restriction requires the paths constituting the functional dependencies not to be prefixes of each other. Under this restriction, k-secrecy is determined as follows. First, compute a set of databases conforming to the given database schema and for which the results of authorized queries are the same as the target database. Next, enumerate databases in the set that return distinct results for the unauthorized query. Then, determine k-secrecy by checking whether there are any such k databases. Technically, the decidability of the *satisfiability of multiple functional dependencies* plays an important role for the enumeration to work. This paper shows that the decidability result of the satisfiability of a single functional dependency [6] can be extended to the multiple case under the non-prefix restriction.

2. Definitions

2.1 Trees

An XML database instance is represented by an *unranked labeled ordered tree*, where the number of each node of a tree is independent of its label. Let $\mathcal{T}_\Sigma$ denote the set of all unranked labeled ordered trees over Σ. The *position* of a node of $t \in \mathcal{T}_\Sigma$ is a sequence of positive integers defined as follows: the position of the root node is ϵ; if the position of a node v is p and v_i is the i-th child of v, then the position of v_i is $p \cdot i$. The nodes and their positions have one-to-one correspondence, so hereafter, we use the term position to mean the node itself. Let $Pos(t)$ be the set of the positions of t. Let $t|_p$ denote the subtree of t at the position p.

Let $\lambda_t(p)$ denote the label of the position p of t. Moreover, let $\tilde{\lambda}_t(p)$ denote the label path from the root to the position p in t, and let $\tilde{\lambda}_t^-(p)$ denote the label path obtained from $\tilde{\lambda}_t(p)$ by removing the leading label. These two notations are useful for expressing a concatenation of label paths concisely, i.e., $\tilde{\lambda}_t(p \cdot p') = \tilde{\lambda}_t(p) \cdot \tilde{\lambda}^-_{t|_p}(p')$.

2.2 Tree automata

We use a *finite unranked tree automaton* (TA) to represent a schema or a set of candidates for the value of the sensitive information. A TA A is a 4-tuple $(Q, \Sigma, \hat{Q}, R)$, where

- Q is a finite set of states,
- Σ is an alphabet,
- $\hat{Q} \subseteq Q$ is a set of initial states, and
- R is a set of transition rules in the form of (q, a, e), where $q \in Q$, $a \in \Sigma$, and e is a regular expression over Q.

Example 2 The following is an example TA A_{H} representing the XML schema supposed in Example 1:

- Q contains Ho, Pa, Na, Di, Ro, Ex, Do, Da, PCDATA;
- Σ contains hospital, patient, name, disease, room, exam, doctor, day;
- $\hat{Q} = \{\mathsf{Ho}\}$; and
- R contains the following rules: (Ho, hospital, Pa*), (Pa, patient, Na $\cdot$ Ro $\cdot$ Di $\cdot$ Ex), (Na, name, PCDATA), (Ro, room, PCDATA), (Di, disease, PCDATA), (Ex, exam, Do $\cdot$ Da), (Do, doctor, PCDATA), (Da, day, PCDATA).

The TA also contains states, symbols, and rules for PCDATA, i.e., string data. In this paper we assume that string data are encoded by trees in some appropriate way [7].

A *(successful) run r^t_A of A on t* is a mapping from $Pos(t)$ to Q with the following properties:

- $r^t_A(\epsilon) \in \hat{Q}$.
- For each position p, if p has n children, there exists a transition rule $(q, a, e) \in R$ such that $r^t_A(p) = q$, $\lambda_t(p) = a$, and $r^t_A(p \cdot 1) r^t_A(p \cdot 2) \cdots r^t_A(p \cdot n)$ is in the string language represented by e.

We say that a tree $t \in \mathcal{T}_\Sigma$ is *accepted* by A if there exists a run of A on t. Let $TL(A)$ denote the tree language *recognized* by A, i.e., the set of trees accepted by A. For $q \in Q$, let $TL(A, q)$ be the tree language recognized by A when the initial state is q. We extend the run to a set P of positions, i.e., $r^t_A(P) = \{r^t_A(p) \mid p \in P\}$. We say A is *unambiguous* if the run r^t_A is unique for each $t \in TL(A)$.

2.3 Queries

We regard queries as tree-to-tree transformation functions. Our verification method requires a query model which preserves inverse recognizability [12]. That is, given a query T and a TA A, a TA which recognizes $\{t' \mid t \in TL(A), T(t') = t\}$ can be constructed. The construction is called *inverse type inference*. Finite compositions of macro tree transducers [10] is one of the query models satisfying the requirement. It is also known that the model is powerful enough to describe many real-world XML transformations.

In this paper, we do not mention a concrete query model, and just assume that queries preserve inverse recognizability.

2.4 Functional dependencies

A *functional dependency* (FD) f is a triple (H, X, Y) where H, X, Y are simple paths over Σ. For a simple path s and a tree $t \in \mathcal{T}_\Sigma$, let $Pos(t, s) = \{p \in Pos(t) \mid \tilde{\lambda}_t(p) = s\}$. $Pos(t, s)$ is the set of positions of t reachable from the root by the path s including the root label. Also, for a position p of t, let $Pos(t, p, s)$ denote the set of positions of t reachable from p by the path s excluding the

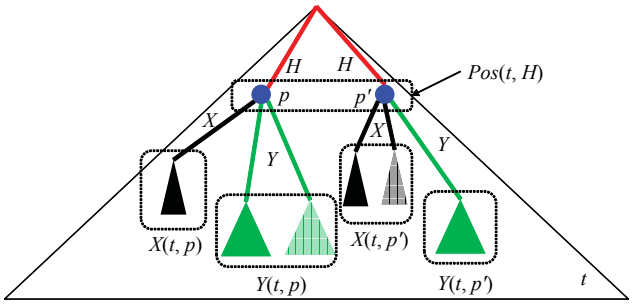

Fig. 4 Definition of an FD.

label of p. Formally, $Pos(t, p, s) = \{p \cdot p' \in Pos(t) \mid \tilde{\lambda}^{-}_{t|_p}(p') = s\}$. We write the set of subtrees of t at positions in $Pos(t, p, s)$ as $s(t, p)$. Given a tree t and an FD f, t *satisfies* f if and only if for any two positions $p, p' \in Pos(t, H)$, $X(t, p) \cap X(t, p') \neq \emptyset$ implies $Y(t, p) \cap Y(t, p') \neq \emptyset$ (see Figure 4). For an FD f, let $TL(f)$ denote the set of trees which satisfy f. FDs $f_1, \ldots, f_N$ are said to be *satisfiable* under a TA A if $TL(A) \cap \bigcap_{i=1}^{N} TL(f_i)$ is not empty.

Our verification method handles a finite number of FDs $f_1, \ldots, f_N$ such that, letting $f_i = (H_i, X_i, Y_i)$,

- for each i $(1 \leq i \leq N)$, neither of X_i nor Y_i is a prefix of the other;
- for each i, j $(1 \leq i, j \leq N, i \neq j)$,
  - if $H_i = H_j$, then none of X_i, X_j, Y_i, and Y_j is a prefix of any of the others;
  - if $H_i \neq H_j$, then neither H_i nor H_j is a prefix of the other.

We refer to this restriction as the *non-prefix restriction*.

Example 3 The two FDs f_1 and f_2 in Example 1 can be represented as follows:

$$f_1 = (\mathsf{hospital} \cdot \mathsf{patient} \cdot \mathsf{exam}, \mathsf{day}, \mathsf{doctor}),$$
$$f_2 = (\mathsf{hospital} \cdot \mathsf{patient}, \mathsf{room}, \mathsf{disease}).$$

Unfortunately, the set $\{f_1, f_2\}$ does not satisfy the non-prefix restriction. However, the following FD f_1' is "equivalent" to f_1 under A_{H} in the sense that $TL(A_{\mathsf{H}}) \cap TL(f_1) = TL(A_{\mathsf{H}}) \cap TL(f_1')$:

$$f_1' = (\mathsf{hospital} \cdot \mathsf{patient}, \mathsf{exam} \cdot \mathsf{day}, \mathsf{exam} \cdot \mathsf{doctor}).$$

Note that the set $\{f_1', f_2\}$ satisfies the non-prefix restriction.

2.5 *k*-secrecy

Let t_G be a target tree to be attacked. We assume that the following information is available to the attackers: the database schema A_G of t_G, authorized queries $T_1, \ldots, T_n$ and their results $T_1(t_G), \ldots, T_n(t_G)$, an unauthorized query T_S, and FDs $f_1, \ldots, f_N$. The sensitive information is $T_S(t_G)$. Suppose that the attacker infers the set L_C of all the candidates for the value of the sensitive information consistent with the above available information, i.e.,

$$L_C = \{T_S(t') \mid t' \in TL(A_G) \cap \bigcap_{i=1}^{N} TL(f_i), \\ T_1(t') = T_1(t_G), \ldots, T_n(t') = T_n(t_G)\}.$$

We say that t_G is *k-secret* (with respect to T_S) if $|L_C| \geq k$.

3. A decision algorithm of *k*-secrecy

Suppose that a target tree t_G to be attacked, authorized $T_1, \ldots, T_n$, unauthorized query T_S, and FDs $f_1, \ldots, f_N$ with the non-prefix restriction are given. In what follows, we show a decision algorithm of k-secrecy of t_G with respect to T_S.

First, we compute

$$L_{INF} = \{t' \in TL(A_G) \mid T_1(t') = T_1(t_G), \ldots, T_n(t') = T_n(t_G)\}$$

by inverse type inference, i.e., we construct a TA A_{INF} such that $L_{INF} = TL(A_{INF})$. Then, letting $A = A_{INF}$, we enumerate candidates for the value of the sensitive information as follows:

- Decide the satisfiability of the FDs $f_1, \ldots, f_N$ under A. If satisfiable, find a tree $u \in TL(A)$ that satisfies the FDs. Also, by inverse type inference, compute the set of trees t such that $T_S(t) = u$. Let A' be a TA such that $TL(A')$ is the difference of $TL(A)$ and the set of such trees t.
- Letting $A = A'$, repeat the above process until k trees are found or the satisfiability check fails. The database is k-secret if and only if k trees are found.

We mainly explain how to check the satisfiability of FDs.

Let A be a TA and $f_1, \ldots, f_N$ be FDs. To check the satisfiability of the FDs under A, construct *path-fixed automata* $A^1_{dv}, \ldots, A^l_{dv}$ explained below. It holds that $TL(A^i_{dv}) \cap TL(A^j_{dv}) = \emptyset$ for each i and j $(1 \leq i, j \leq l, i \neq j)$, and $TL(A) = \bigcup_{1 \leq i \leq l} TL(A^i_{dv})$. Satisfiability check is done for each of these path-fixed automata. That is, FDs $f_1, \ldots, f_N$ are satisfiable under A if and only if there is some TA A^i_{dv} under which FDs $f_1, \ldots, f_N$ are satisfiable.

A TA A is *path fixed* with respect to FDs $f_1, \ldots, f_N$ if A is unambiguous and satisfies the following three conditions for each $f = (H, X, Y) \in \{f_1 \ldots, f_N\}$:

(1) $\forall t, t' \in TL(A).r^I_A(Pos(t, H)) = r^{I'}_A(Pos(t', H))$.

(2) $\forall t, t' \in TL(A).\forall p \in Pos(t, H).\forall p' \in Pos(t', H)$.
$(r^I_A(p) = r^{I'}_A(p') \Rightarrow \forall Z \in \{X, Y\}$.
$r^I_A(Pos(t, p, Z)) = r^{I'}_A(Pos(t', p', Z)))$.

(3) $\forall t \in TL(A).\forall Z \in \{X, Y\}$.
$\forall p, p' \in Pos(t, HZ).r^I_A(p) \neq r^I_A(p') \Rightarrow t|_p \neq t|_{p'}$.

This is an extension of the notion of *f-path fixity* introduced in [6] to multiple FDs. The first condition means that for every tree in $TL(A)$, the set of states assigned to the positions in $Pos(t, H)$ is fixed. The fixed set is denoted by Q^A_H. The second condition means that for any tree in $TL(A)$ and any position p in $Pos(t, H)$, the set of states assigned to the positions in $Pos(t, p, X)$ (resp. $Pos(t, p, Y)$) is fixed on the state assigned to the position p. For each $q_h \in Q^A_H$, the fixed sets are denoted by $Q^A_{q_h,X}$ and $Q^A_{q_h,Y}$, respectively. The third condition means that for any tree in $TL(A)$ and any two positions in either of $Pos(t, HX)$ or $Pos(t, HY)$, if the states assigned to the positions are distinct, then so are the subtrees at the positions. Let $Q^A_{HX} = \bigcup_{q_h \in Q^A_H} Q^A_{q_h,X}$.

Fix $f = (H, X, Y)$ in $\{f_1, \ldots, f_n\}$ and let $q_x \in Q^A_{HX}$. Consider a subset Q' of Q^A_H such that for any distinct q_{h1} and q_{h2} in Q',

- $q_x \in Q^A_{q_{h1},X} \cap Q^A_{q_{h2},X}$, and
- $Q^A_{q_{h1},Y} \cap Q^A_{q_{h2},Y} = \emptyset$.

For q_x, such Q' is not unique. Let $k(q_x)$ denote the maximum size of such Q'. In order to satisfy $f = (H, X, Y)$, we have to assign distinct trees in $TL(A, q_x)$ to their distinct ancestor states in Q' since $Q^A_{q_{h1},Y} \cap Q^A_{q_{h2},Y} = \emptyset$. Hence, it can be shown that f is satisfiable if and only if $|TL(A, q_x)| \geq k(q_x)$ for all $q_x \in Q^A_{HX}$. Satisfiability can be checked independently because of the non-

prefix restriction. Therefore we have the following theorem.

Theorem 1 Let $f_1,\ldots, f_N$ be FDs satisfying non-prefix restriction, and A be a general TA. Satisfiability of $f_1,\ldots, f_N$ under A is decidable.

Using the decidability result of satisfiability of FDs, we can show the decidability of k-secrecy.

Theorem 2 k-secrecy against inference attacks using FDs is decidable, provided that the FDs satisfy the non-prefix restriction. Moreover, if k-secret, $u_1, \ldots, u_k \in L_{INF}$ such that $T_S(u_i) \neq T_S(u_j)$ for any i and j $(1 \leq i, j \leq k, i \neq j)$ are computable.

4. Related Work

Inference attacks have been one of the most well-known threats on databases for the past few decades. On relational databases, aggregate functions can be used for inferring sensitive information [3]. Disclosure Monitor [2] is a part of a relational database management system that monitors information disclosure by inference attacks. Roughly speaking, Disclosure Monitor keeps track of users' knowledge obtained by queries issued so far. When a user issues a new query, Disclosure Monitor determines whether the result of the new query with the current knowledge of the user disclose the sensitive information. According to the determination result, Disclosure Monitor decides whether the new query should be allowed or not. Several stronger security definitions [4], [11] require that authorized views and the answers of them do not change the probability distribution of possible secrets. As for XML databases, there have been a few studies on secure view publishing [5], [14].

Security against inference attacks is often discussed in the context of privacy protection. k-anonymity [13] is one of the most famous security criteria, which assumes *linking attacks* to privacy data in multiple tables. A set of attributes that can be useful for identifying individuals is called a pseudo-identifier. The concept of k-anonymity is based on the idea that a database is safe if it contains many corresponding tuples for each possible value of a pseudo-identifier. Another famous criterion is l-diversity [9]. It is based on the idea that a database is safe if it contains many candidates for values of sensitive information for each possible value of a pseudo-identifier. Our notion of k-secrecy is similar to the notion of l-diversity but differs in that our model assumes attackers infers *all the candidates* for the value of sensitive information *consistent with the information available to the attackers*. That is, it is assumed that attackers can perform more than linking attacks. Our result is therefore useful for guaranteeing higher secrecy than l-diversity.

Research on inference attacks is closely related to research on incomplete information because an attacker's knowledge is considered as incomplete information on the sensitive information. Conditional tables [8] are a simple but powerful representation of incomplete relational databases. In conditional tables, unknown values are represented by variables, and the domains of variables and the existence of tuples are specified by conditional expressions. Actually, to keep track of the user's knowledge, Disclosure Monitor uses a data structure similar to conditional tables. As for XML databases, incomplete trees were proposed [1]. They can handle trees with data values, but only a limited number of tree shapes. In our formulation, data values are assumed to be encoded by trees. Therefore, we can adopt finite tree automata as a representation of incomplete information, which have good closure properties, although comparisons between data values are limited.

5. Conclusion

In this paper, we have shown that k-secrecy against inference attacks using multiple FDs on XML databases is decidable, provided the FDs satisfy the non-prefix restriction. As demonstrated in Example 1, inference using multiple FDs is strictly more powerful than inference using a single FD. Our result shows that the risk by multiple FDs is detectable, while it is not necessarily detected by existing methods.

The non-prefix restriction is critical for our decision procedure to work correctly. For example, consider the following two FDs: $f_1 = (H_1, X_1, Y_1)$ and $f_2 = (H_2, X_2, Y_2)$. Without the restriction, H_2X_2 might be a prefix of H_1. The number of possible subtrees at H_2X_2 could not be independent of the numbers of possible subtrees at H_1X_1 and H_1Y_1. This means that it is impossible to decide the satisfiability for each FD independently. Our future work will include relaxing the non-prefix restriction so that k-secrecy is still decidable.

References

[1] Abiteboul, S., Segoufin, L. and Vianu, V.: Representing and Querying XML with Incomplete Information, *Proceedings of the Twentieth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems* (2001).
[2] Brodsky, A., Farkas, C. and Jajodia, S.: Secure databases: Constraints, inference channels, and monitoring disclosures, *IEEE Transactions on Knowledge and Data Engineering*, Vol. 12, No. 6, pp. 900–919 (2000).
[3] Denning, D. E. R.: *Cryptography and Data Security*, Addison-Wesley (1982).
[4] Deutsch, A. and Papakonstantinou, Y.: Privacy in Database Publishing, *Proceedings of the 10th International Conference on Database Theory*, pp. 230–245 (2005).
[5] Fan, W., Chan, C. Y. and Garofalakis, M. N.: Secure XML Querying with Security Views, *Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data*, pp. 587–598 (2004).
[6] Hashimoto, K., Kawai, H., Ishihara, Y. and Fujiwara, T.: 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, pp. 1365–1374 (2012).
[7] Hashimoto, K., Sakano, K., Takasuka, F., Ishihara, Y. and Fujiwara, T.: Verification of the Security against Inference Attacks on XML Databases, *IEICE Transactions on Information and Systems*, Vol. E92-D, No. 5, pp. 1022–1032 (2009).
[8] Imielinski, T. and Jr., W. L.: Incomplete Information in Relational Databases, *Journal of the ACM*, Vol. 31, No. 4, pp. 761–791 (1984).
[9] Machanavajjhala, A., Gehrke, J., Kifer, D. and Venkitasubramaniam, M.: ℓ-Diversity: Privacy Beyond k-Anonymity, *Proceedings of the 22nd International Conference on Data Engineering*, p. 24 (2006).
[10] Maneth, S., Berlea, A., Perst, T. and Seidl, H.: XML type checking with macro tree transducers, *Proceedings of the 24th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pp. 283–294 (2005).
[11] Miklau, G. and Suciu, D.: A formal analysis of information disclosure in data exchange, *Journal of Computer and System Sciences*, Vol. 73, No. 3, pp. 507–534 (2007).
[12] Suciu, D.: The XML Typechecking Problem, *SIGMOD Record*, Vol. 31, No. 1, pp. 89–96 (2002).
[13] Sweeney, L.: k-anonymity: A model for protecting privacy, *International Journal on Uncertainty, Fuzziness and Knowledge-based Systems*, Vol. 10, No. 5, pp. 557–570 (2002).
[14] Yang, X. and Li, C.: Secure XML Publishing without Information Leakage in the Presence of Data Inference, *Proceedings of the 30th International Conference on Very Large Data Bases*, pp. 96–107 (2004).