PAPER Special Section on Formal Approach

Decidability of the Security against Inference Attacks Using a Functional Dependency on XML Databases

Kenji HASHIMOTO[†a)], ***Member***, **Hiroto KAWAI**[††], ***Nonmember***, **Yasunori ISHIHARA**[†††], ***and*** **Toru FUJIWARA**[†††], ***Members***

SUMMARY This paper discusses verification of the security against inference attacks on XML databases in the presence of a functional dependency. So far, we have provided the verification method for k-secrecy, which is a metric for the security against inference attacks on databases. Intuitively, k-secrecy means that the number of candidates of sensitive data (i.e., the result of unauthorized query) of a given database instance cannot be narrowed down to $k-1$ by using available information such as authorized queries and their results. In this paper, we consider a functional dependency on database instances as one of the available information. Functional dependencies help attackers to reduce the number of the candidates for the sensitive information. The verification method we have provided cannot be naively extended to the k-secrecy problem with a functional dependency. The method requires that the candidate set can be captured by a tree automaton, but the candidate set when a functional dependency is considered cannot be always captured by any tree automaton. We show that the ∞-secrecy problem in the presence of a functional dependency is decidable when a given unauthorized query is represented by a deterministic topdown tree transducer, without explicitly computing the candidate set.

key words: *XML database, inference attack, security, verification, functional dependency*

1. Introduction

Nowadays, many people and organizations have a growing interest in data security. For a database system to be secure, secrecy, integrity, and availability of data must be achieved appropriately with respect to a given security policy. View mechanisms has played a principal role in achieving the secrecy of database systems. Views can avoid direct accesses to sensitive information in the database. However, there still is a possibility of indirect accesses, i.e., the sensitive information can be inferred using authorized views and/or general domain knowledge. Such indirect accesses are called inference attacks.

Example 1: We show an example of inference attacks on XML databases, which are getting used actively in many organizations recently. We consider a document D valid against the following schema A:

Manuscript received July 18, 2011.
Manuscript revised November 13, 2011.

[†]The author is with the Graduate School of Information Science, Nara Institute of Science and Technology, Ikoma-shi, 630–0192 Japan.
[††]The author is with NTT WEST Corporation, Osaka-shi, 540–0007 Japan.
[†††]The authors are with the Graduate School of Information Science and Technology, Osaka University, Suita-shi, 565–0871 Japan.
a) E-mail: k-hasimt@is.naist.jp

DOI: 10.1587/transinf.E95.D.1365

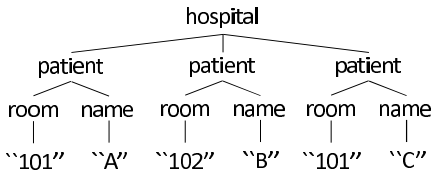


Fig. 1 The result $T_1(D)$.

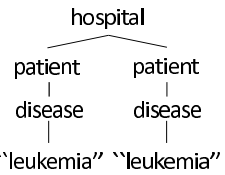


Fig. 2 The result $T_2(D)$.

```
<!ELEMENT hospital (patient*)>
<!ELEMENT patient (room, name, disease)>
<!ELEMENT room (#PCDATA)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT disease (#PCDATA)>
```

A `hospital` has zero or more `patient`s. Each `patient` has a `room`, a `name`, and a `disease`. Since `#PCDATA` roughly represents string data, each of `room`, `name`, and `disease` has a string as its value.

We consider the following two authorized queries T_1 and T_2 on D: Query T_1 extracts, for each `patient`, the `room` number in which the `patient` is and the name of the `patient`. We assume that the result $T_1(D)$ is the tree in Fig. 1. Query T_2 extracts, for each `patient` who has "leukemia", the `patient`'s `disease`. We assume that the result $T_2(D)$ is the tree in Fig. 2.

Now, let T_s denote the query which extracts the name of the disease each patient has, and assume that $T_s(D)$ is sensitive data. Then, from the authorized queries T_1 and T_2, and the results $T_1(D)$ and $T_2(D)$, it turns out that there exist candidates for the value of $T_s(D)$ as shown in Fig 3. Here, we assume that the patients who are in the same room have the same disease. Then, from $T_1(D)$, we know that patients 'A' and 'C' are in the same room. Thus, 'A' and 'C' have the same disease. From $T_2(D)$, the number of the patients with leukemia are two. Therefore, we find that patients 'A' and 'C' have leukemia, and patient 'B' has some disease other than leukemia. □

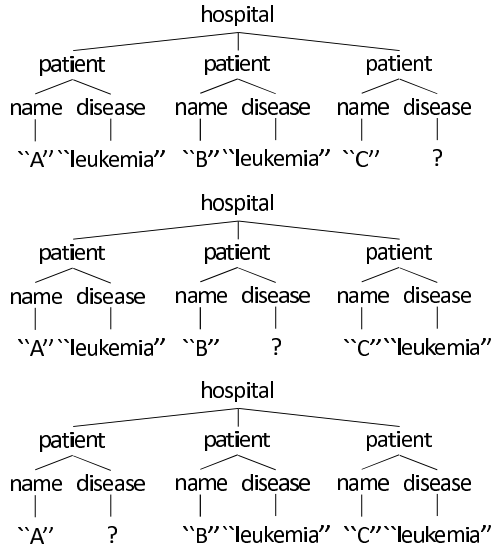


Fig. 3 Candidates for sensitive data.

Inference attacks, in particular on relational databases, have been studied for a few decades. Inference attacks on relational databases using aggregate functions are one of the most famous threats [1]. Also, several sophisticated formalizations of the security have also been proposed, e.g., the ones based on information theory [2], [3] and the one based on rough set theory [4]. Moreover, Deutsch and Papakonstantinuou [5] and Miklau and Suciu [6] proposed stronger security definitions where the probability distribution of possible secrets does not change before and after authorized views and the answers of them are given. Recently, the security against inference attacks is often discussed in the context of privacy protection. Machanavajjhala et al. proposed the notion of ℓ-diversity [7], which means that for each person, there are at least ℓ possible values (or ℓ bits in entropy, etc.) for the person's sensitive information.

We have proposed a verification method for *k-secrecy*, a security criteria against inference attacks on XML databases [8]. The concept of k-secrecy is as follows: Let t be a target XML document to be attacked. Suppose that the following information is available to the attacker:

- the schema A_G of t,
- authorized queries (i.e., view definitions) $T_1, \ldots, T_n$,
- results of the authorized queries on t (i.e., views) $T_1(t), \ldots, T_n(t)$, and
- a query T_s to retrieve the sensitive information in t.

Also, suppose that the attacker infers the set C of all the candidates for the value of the sensitive information consistent with the information available to the attacker. That is,

$$C = \{T_s(t') \mid t' \in TL(A_G), \\ T_1(t') = T_1(t), \ldots, T_n(t') = T_n(t)\},$$

where $TL(A_G)$ is the set of the XML documents valid against A_G. In particular, t is ∞-*secret* if $|C|$ is infinite. However, in our previous work [8], the functional dependencies are not considered. For instance, in Example 1, the knowledge that patients who are in the same room have the same disease can be represented by a functional dependency. Functional dependencies help attackers to reduce the number of the candidates for the value of sensitive information.

On the other hand, Yang and Li discussed XML data publishing secure against inference attack considering functional dependencies [9]. Attackers are supposed to use functional dependencies as well as schema information and the view XML document. Then, an algorithm for finding a maximal view document without allowing successful inference attacks is proposed. They focus on data publishing, while we discuss verification of the security against inference attack.

We consider the verification for ∞-secrecy in the presence of a functional dependency. The verification method in our previous work [8] cannot be naively extended to this problem. In the method, first, the candidate set of the original database instances from authorized queries and their results is computed. Without functional dependencies on the database instance, the set can be represented by a finite tree automaton (TA), that is, the set is regular. On the other hand, the set of the candidates of the original database instance satisfying a simple functional dependency is not always regular. The previous verification method requires the regularity of the candidate set.

In this paper, as the first step, we show that ∞-secrecy problem in the presence of a simple functional dependency is decidable when an unauthorized query is represented by a deterministic topdown tree transducer. Our decision algorithm consists of three steps. First, we compute the candidate set for the original database instance from authorized queries and their results in the same way as our previous method [8]. Though the set is regular, it might include candidates which do not satisfy the functional dependency f. Second, we divide the set into a finite number of sets each of which can be captured by an *f-path fixed* tree automaton. Intuitively, every tree accepted by an f-path fixed TA has the same fixed parts in the sense of the satisfaction of f. This allows us to check whether there is some tree accepted by an f-path fixed TA which satisfies f more simply. Lastly, we check, for each f-path fixed TA A, whether the image of L through the unauthorized query T_s is infinite where L is the set of trees each of which is accepted by A and satisfies f. We provide a necessary and sufficient condition for such infiniteness, which does not need to compute L explicitly.

The rest of the paper is organized as follows. Section 2 introduces models of XML documents, schemas, queries, and functional dependencies. Section 3 defines k-secrecy problem in the presence of a functional dependency. Section 4 presents the decidability result of the k-secrecy problem. In Sect. 5, we discuss the difficulty of applying our algorithm to ∞-SIAF for more powerful query model such as bottom-up tree transducers. Section 6 summarizes the paper.

2. Models

In this section, first, ordinary models of XML documents and schemas are introduced. Then, a model of queries is presented. Lastly, a functional dependency is introduced.

2.1 XML Documents

An XML document is represented by an *unranked* labeled ordered tree, i.e., a labeled ordered tree where the number of the children of a node is not bounded. Let Σ be an alphabet including special symbols \$ and $\sharp$, which will be explained in Sect. 2.3. Let $\mathcal{T}_\Sigma$ denote the set of the unranked labeled ordered trees over Σ, i.e., every node of the trees in $\mathcal{T}_\Sigma$ is labeled with a symbol in Σ. By $a(t_1 \cdots t_n)(n \geq 0)$, we mean the tree whose root is labeled with a and has immediate subtrees $t_1, \ldots, t_n$ from the left in the order. We often write a instead of $a()$. The *position* of a node of t is 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 $p \cdot i$. Let $Pos(t)$ be the set of the positions of nodes in t. For a position p of t, let $Anc_t(p)$ be the set of the ancestor positions of p. Also, for each position p in $t \in \mathcal{T}_\Sigma$, let $\lambda_t(p)$ denote the label of the node on the position p of t. 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. Let $t|_p$ denote the subtree of t at the position p, and $t[p \leftarrow t']$ denote the tree obtained from t by substituting the subtree at p with t'.

As many theoretical studies do, we would like to avoid distinguishing text nodes (i.e., nodes with string values) with element nodes (i.e., nodes with tag names). To do this, we simulate such a node by an element node named `string` and its children which are also element nodes labeled by the constituent characters. Also, in this paper, numerical values cannot be handled directly; they are treated as strings of digits.

2.2 Sets of XML Documents

In this paper, we use a *tree automaton* (TA) to represent schemas or sets of candidates for the value of the sensitive information.

First, we define a regular expression. A regular expression (RE) over an alphabet Σ consists of constants $\emptyset$ (empty set), ϵ (empty sequence), and the symbols in Σ, and operators $\cdot$ (concatenation), $*$ (zero or more occurrences), + (one or more occurrences), | (disjunction), and & (interleaving). The concatenation operator is often omitted as usual. The string language represented by a regular expression e is denoted by $L(e)$.

Next, we define an (unranked) TA over Σ. A TA A is a 4-tuple (Q, Σ, Q^i, R), where

- Q is a finite set of states,
- Σ is an alphabet,
- $Q^i \subseteq Q$ 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 an RE over Q.

We introduce the run of a TA on a tree. Let $A = (Q, \Sigma, Q^i, R)$ be an TA and $t \in \mathcal{T}_\Sigma$. A *(successful) run* r_A^t *of* A *on* t is a mapping from $Pos(t)$ to Q with the following properties:

- $r_A^t(\epsilon) \in Q^i$.
- For each position p, if the node of p has n children, there exists a transition rule $(q, a, e) \in R$ such that
  - $r_A^t(p) = q$,
  - $\lambda_t(p) = a$, and
  - $r_A^t(p \cdot 1) r_A^t(p \cdot 2) \cdots r_A^t(p \cdot n) \in L(e)$.

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 set of trees accepted by A, i.e., the tree language *recognized* 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., $r_A^t(P) = \{r_A^t(p) \mid p \in P\}$. For $t \in TL(A)$, we define the state path $\tilde{r}_A^t(p)$ to p on $\tilde{r}_A^t$ as follows: $\tilde{r}_A^t(\epsilon) = r_A^t(\epsilon)$; $\tilde{r}_A^t(p \cdot i) = \tilde{r}_A^t(p) r_A^t(p \cdot i)$. We say A is *unambiguous* if the run r_A^t is unique for each $t \in TL(A)$. We say A is *bottom-up deterministic* if $TL(A, q_1) \cap TL(A, q_2) = \emptyset$ for any two states $q_1, q_2 \in Q$.

2.3 Queries

We require a query model for which type inference [10] can be done. Hence, as in our previous work [8], we adopt *(deterministic) bottom-up relabeling tree transducers* (BRTTs), *(deterministic) top-down relabeling tree transducers* (TRTTs), and *(deterministic) deleting tree transducer* (DTTs). A BRTT and a TRTT relabel nodes dependently on their descendants and ancestors, respectively. A DTT deletes all nodes labeled with a special symbol $\sharp$, and all subtrees rooted by nodes labeled with another special symbol \$. A TRTT and a DTT are restricted ones of top-down tree transducers [11] which can only relabel and delete nodes of an input tree respectively. A BRTT is a straightforward extension of a bottom-up tree automaton. In this paper, a query is assumed to be a composition of some of BRTTs and/or TRTTs, followed by a DTT. For tree transductions T^R and T^D, let $T^D \circ T^R$ denote T^R composed with T^D, i.e., $T^D \circ T^R(t) = T^D(T^R(t))$ for any tree t. Consequently, a query in our model can filter and/or relabel nodes selected by expressions in the class $XP^{\{[\ ],*,//\}}$ [12], which is a fragment of XPath consisting of node test, the child axis, the descendant axis, wildcards, and predicates. Moreover, our model can represent some, but not all, queries which process nodes selected by expressions with the sibling axis. The query cannot reconstruct the input tree with its document order changed, add new nodes to the input tree, or copy nodes of the input tree.

As a more powerful query model, an m-fold composition of macro tree transducers (m-MTT) [13] has sufficient

expressive power where m is a constant. This model enables insertion, copying, and concatenation of trees. However, we conjecture that these abilities make analysis for ∞-secrecy complicated. In this paper, as the first step, we shall tackle analysis for ∞-secrecy with respect to queries expressed by a composition of some of BRTTs and/or TRTTs, followed by a DTT.

2.3.1 Bottom-Up Relabeling Tree Transducers (BRTTs)

A BRTT T^{B} is a 4-tuple $(Q, \Sigma, \hat{q}, F^{\mathrm{B}})$, where

- Q is a finite set of states,
- Σ is an alphabet,
- $\hat{q} \in Q$ is the final state, and
- F^{B} is a set of transformation rules in the form of $a(e) \to q'(a')$, where $q' \in Q$, $a, a' \in \Sigma$, and e is an RE over Q. If F^{B} contains two distinct rules $a(e_1) \to q'_1(a'_1)$ and $a(e_2) \to q'_2(a'_2)$, then $L(e_1) \cap L(e_2)$ must be the empty language.

Let t_s be a tree in $\mathcal{T}_{\Sigma \cup Q}$ and $t_p = a(q_1(t_1) \cdots q_n(t_n))$ be a subtree of t_s, where $a \in \Sigma$, $t_1, \ldots, t_n \in \mathcal{T}_\Sigma$ and $q_1, \ldots, q_n \in Q$. If $(a(e) \to q'(a')) \in F^{\mathrm{B}}$ and $q_1 \cdots q_n \in L(e)$, then T^{B} can move in one step from t_s to t'_s such that t'_s is obtained from t_s by replacing t_p to $q'(a'(t_1 \cdots t_n))$. A tree $t \in \mathcal{T}_\Sigma$ is *transformed* into $t' \in \mathcal{T}_\Sigma$ by T^{B} if T^{B} can move from t eventually to $\hat{q}(t')$. Then, let $T^{\mathrm{B}}(t)$ denote the tree t'.

2.3.2 Top-Down Relabeling Tree Transducers (TRTTs)

A TRTT T^{T} is a 4-tuple $(Q, \Sigma, \hat{q}, F^{\mathrm{T}})$, where

- Q is a finite set of states,
- Σ is an alphabet,
- $\hat{q} \in Q$ is the initial state, and
- F^{T} is a set of transformation rules in the form of $q(a) \to a'(q')$, where $q, q' \in Q$ and $a, a' \in \Sigma$. For each pair of q and a, there must be at most one rule in F^{T} whose left-hand side is $q(a)$.

Let t_s be a tree in $\mathcal{T}_{\Sigma \cup Q}$ and $t_q = q(a(t_1 \cdots t_n))$ be a subtree of t_s, where $a \in \Sigma$, $t_1, \ldots, t_n \in \mathcal{T}_\Sigma$ and $q \in Q$. If $(q(a) \to a'(q')) \in F^{\mathrm{T}}$, then T^{T} can move in one step from t_s to t'_s such that t'_s is obtained from t_s by replacing t_q with $a'(q'(t_1) \cdots q'(t_n))$. Note that by definition a state associated with a leaf node will disappear after the next transformation step because a leaf node is a node with the empty sequence of child subtrees. A tree $t \in \mathcal{T}_\Sigma$ is *transformed* into $t' \in \mathcal{T}_\Sigma$ by T^{T} if T^{T} can move from $\hat{q}(t)$ eventually to $t' \in \mathcal{T}_\Sigma$. Then, let $T^{\mathrm{T}}(t)$ denote the tree t'.

2.3.3 Deleting Tree Transducers (DTTs)

A DTT T^{D} deletes $\sharp$-labeled nodes of the input tree and subtrees rooted by \$-labeled nodes of the tree, traversing in a top-down manner. For example, $t = a(b\sharp(de)b(\sharp f\$)\$(de))$ is transformed to $t' = a(bdeb(f))$ by T^{D}. Exceptionally, the root of the input tree is never deleted to ensure that the output is a tree.

2.3.4 Assumptions

We introduce the following two assumptions on our query model. First, we assume for simplicity that each authorized query T_a is total to $TL(A_G)$, where A_G is the schema of the target XML document D. That is, for each $t \in TL(A_G)$, there is $T_a(t) \in \mathcal{T}_\Sigma$. Second, we assume that no constituent relabeling tree transducers of a query T_s to retrieve the sensitive information relabels a node with $\sharp$. Therefore, T_s can delete some subtrees of the input tree, but cannot delete only internal nodes.

2.4 Functional Dependencies (FDs)

An FD is a triple of simple path expressions. The simple path expression is a sequence over Σ. This class is a subclass of XPath [14]. The semantics of a simple path expression s over a tree t is defined by $Pos(t, s)$:

$$Pos(t, s) := \{p \in Pos(t) \mid \tilde{\lambda}_t(p) = s\}.$$

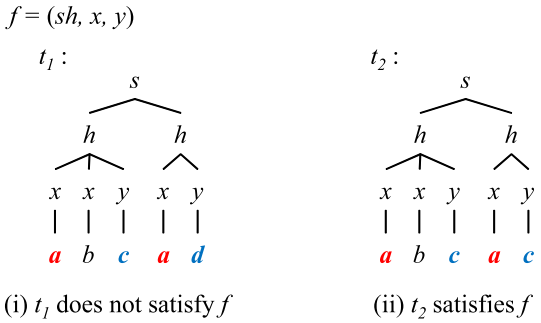
$Pos(t, s)$ is the set of positions of t reachable from the root by the path s including the root label. Also, for a position p of t, let $Pos(t, p, s)$ denote the set of positions of t reachable from p by the path s excluding the label of the node at p. Formally,

$$Pos(t, p, s) := \{p \cdot p' \in Pos(t) \mid \tilde{\lambda}^-_{t|_p}(p') = s\}.$$

In addition, we write the set of subtrees of t at positions in $Pos(t, s)$ (resp. $Pos(t, p, s)$) as $s(t)$ (resp. $s(t, p)$).

An FD f is a triple (H, X, Y) where H, X, Y are simple path expressions over Σ, and neither of X nor Y is a prefix of the other. Given a tree t and an FD f, t is said to satisfy f if and only if for any two positions $p, p' \in Pos(t, H)$, $X(t, p) \cap X(t, p') \neq \emptyset \Rightarrow Y(t, p) \cap Y(t, p') \neq \emptyset$. For an FD f, let $TL(f)$ denote the set of trees which satisfy f.

For example, let f be a functional dependency (sh, x, y). In Fig. 4, the tree (i) does not satisfy f because both subtrees rooted by the nodes labeled with h include the subtree $x(a)$ but they do not include any common subtree rooted by nodes labeled with y. On the other hand, the tree (ii) satisfies f because both subtrees rooted by the nodes labeled with h include the common subtrees rooted



(i) t_1 does not satisfy f (ii) t_2 satisfies f

Fig. 4 Functional dependency.

by nodes labeled with x and y, respectively. In addition, the functional dependency in Example 1 can be represented as ($\texttt{hospital} \cdot \texttt{patient}, \texttt{room}, \texttt{disease}$).

So far, Arenas and Libkin's [15] and Yu and Jagadish [16] have proposed definitions of functional dependencies on XML. Our definition of FDs differs from them, but is relatively close to Yu and Jagadish's. In Yu and Jagadish's, the path H in our FD expression is called the pivot path. However, as the semantics of agreement between $X(t, p)$ and $X(t, p')$ (or $Y(t, p)$ and $Y(t, p')$) for any two nodes specified by the pivot path, Yu and Jagadish's adopts $X(t, p) = X(t, p')$ (or $Y(t, p) = Y(t, p')$), respectively. On the other hand, we adopt $X(t, p) \cap X(t, p') \neq \emptyset$ (or $Y(t, p) \cap Y(t, p') \neq \emptyset$), respectively, which is closer to the semantics of equality test of sets of nodes in XPath 1.0 [14]. In addition, Yu and Jagadish's ignores the document order on the equality of trees, but we consider it.

3. k-Secrecy Problem in the Presence of a Functional Dependency

We consider a decision problem for k-secrecy (or ∞-secrecy) against inference attacks in the presence of a functional dependency, abbreviated as k-SIAF (or ∞-SIAF).

Definition 1: The database schema A_G of the instance t, authorized queries $T_1, \ldots, T_n$, results $T_1(t), \ldots, T_n(t)$ of the authorized queries, unauthorized query T_s, and a functional dependency f are given. Then, suppose that

$$C = \{T_s(t') \mid t' \in TL(A_G) \cap TL(f), \\ T_1(t') = T_1(t), \ldots, T_n(t') = T_n(t)\}.$$

If $|C| \geq k$ then we say that t is k-secret with respect to $(\{T_1, \ldots, T_n\}, T_s, f)$. In particular, if C is infinite then we say that t is ∞-secret with respect to $(\{T_1, \ldots, T_n\}, T_s, f)$.

In this paper, we show the decidability of ∞-SIAF. It is one of our future work to investigate k-SIAF for any positive integer $k > 1$.

As stated in Sect. 1, the verification method in our previous work [8], which does not consider functional dependencies, cannot be naively extended to ∞-SIAF. It is because the set of the candidates of the original database instance, which satisfy a functional dependency, is not always regular. For example, consider an FD $f = (sh, x, y)$ and a TA A which has the following five transition rules: $(S, s, HH), (H, h, XY), (X, x, C), (Y, y, C^*), (C, c, \epsilon)$ where S is the initial state. Then, $TL(A) \cap TL(f)$ is not regular because each tree in $TL(A) \cap TL(f)$ has the same two subtrees the roots of which are labeled with h. Thus, $TL(A) \cap TL(f)$ cannot be represented by any TA.

In this paper, we give a method to verify the ∞-secrecy without explicitly computing the set of the candidates of the original database instance satisfying functional dependencies.

4. Decidability Result of ∞-SIAF

Here, we show that ∞-SIAF is decidable when the unauthorized query T_s is restricted to a composition of TRTTs and a DTT. Since TRTTs are closed under composition, we deal with only a composition of one TRTT and a DTT.

4.1 Overview of Our Decision Algorithm

Our decision algorithm for ∞-secrecy with an FD consists of three steps:

1. Construct a TA A_a which recognizes $\bigcap_{i=1}^{n} T_i^{-1}(T_i(t)) \cap TL(A_G)$, where T_i^{-1} is the inverse of T_i, that is, $T_i^{-1}(t') = \{t \mid T_i(t) = t'\}$.
2. Divide A_a into a finite set $\mathcal{A}$ of *f-path fixed* TAs. (The definition of f-path fixity of a TA will be given in Sect. 4.2.)
3. Decide whether there is some TA $A \in \mathcal{A}$ such that $TL(A) \cap TL(f) \neq \emptyset$ and $T_s(TL(A))$ is infinite.

In Step 1, given each T_i and $T_i(t)$ ($1 \leq i \leq n$), a TA A_a^i which recognizes $T_i^{-1}(T_i(t))$ can be constructed from $T_i(t)$ and T_i by backward type inference. Note that T_i has the backward regularity preserving property, i.e., for any regular tree language L, $T_i^{-1}(L)$ is also a regular tree language. Finally, A_a can be obtained by constructing the intersection TA of $A_a^1, \ldots, A_a^n$ and A_G.

In Steps 2 and 3, it is decided whether $\{T_s(t) \mid t \in TL(A_a) \cap TL(f)\}$ is infinite. It is complicated to decide it directly from T_s, A_a, and f because trees accepted by A_a have various structure related on f. Our strategy for the decision is as follows: Divide the given TA A_a to a finite set $\mathcal{A}$ of f-path fixed TAs, and then, for each $A_i \in \mathcal{A}$, check if A_i accepts some tree t satisfying f and $T_s(TL(A_i))$ is infinite. Intuitively, the parts related on f of runs on accepted trees of an f-path fixed TAs are identical. This property makes the satisfiability check of f on the TAs relatively simple.

4.2 f-Path Fixity

Before giving details of our decision algorithm for ∞-SIAF, we first give the definition of f-path fixity of a TA. For an FD $f = (H, X, Y)$, we say that A is *f-path fixed* if and only if A is unambiguous and satisfies the following three conditions:

(1) $\forall t, t' \in TL(A).r_A^t(Pos(t, H)) = r_A^{t'}(Pos(t', H))$,

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

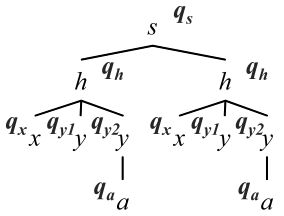
(3) $\forall t \in TL(A).\forall Z \in \{X, Y\}.\forall p, p' \in Pos(t, HZ).$
$r_A^t(p) \neq r_A^t(p') \Rightarrow t|_p \neq t|_{p'}$.

The condition (1) means that for every tree in $TL(A)$, the set of states assigned to nodes at the positions in $Pos(t, H)$

$f = (sh, x, y)$

(i) A_1: f-path fixed

(q_s, s, q_h^+),
$(q_h, h, q_x q_{y1} q_{y2})$,
(q_x, x, ε),
(q_{y1}, y, ε),
(q_{y2}, y, q_a),
(q_a, a, ε)



(ii) A_2: Not f-path fixed

(q_s, s, q_h^+),
$(q_h, h, q_x q_{y1} | q_x q_{y2})$,
(q_x, x, ε),
(q_{y1}, y, ε),
(q_{y2}, y, q_a),
(q_a, a, ε)

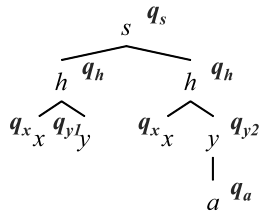


Fig. 5 f-path fixity.

is fixed. The fixed set is denoted by Q^A_H. The condition (2) means that for any tree in $TL(A)$ and any position p in $Pos(t, H)$, the set of states assigned to nodes at the positions in $Pos(t, p, X)$ (resp. $Pos(t, p, Y)$) is fixed on the state assigned to the node at 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 condition (3) 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 nodes at the positions are distinct, then so are the subtrees at the positions.

For example, let $f = (sh, x, y)$. Then, consider the TA A_1 which has the transition rules listed in Fig. 5 (i). For every tree $t \in TL(A_1)$, $r^t_{A_1}(Pos(t, sh)) = \{q_h\}$, and for each position p in $Pos(t, sh)$, $r^t_{A_1}(Pos(t, p, x)) = \{q_x\}$ and $r^t_{A_1}(Pos(t, p, y)) = \{q_{y1}, q_{y2}\}$. Thus, A_1 satisfies the conditions (1) and (2) of the f-path fixity. Also, A_1 satisfies the condition (3) because $TL(A_1, q_{y1}) \cap TL(A_1, q_{y2}) = \emptyset$. Thus, A_1 is f-path fixed. On the other hand, consider the TA A_2 which has the transition rules listed in Fig. 5 (ii). A_2 accepts the tree $t' = s(h(x, y), h(x, y(a)))$. Then, we have that $Pos(t', sh) = \{1, 2\}$, $r^{t'}_{A_2}(Pos(t', 1, y)) = \{q_{y1}\}$ and $r^{t'}_{A_2}(Pos(t', 2, y)) = \{q_{y2}\}$, and thus the condition (2) is not satisfied. Therefore, A_2 is not f-path fixed.

We give some notations on an f-path fixed TA A where $f = (H, X, Y)$. Let $Q^A_{HX} = \bigcup_{q_h \in Q^A_H} Q^A_{q_h,X}$ and $Q^A_{HY} = \bigcup_{q_h \in Q^A_H} Q^A_{q_h,Y}$. For a tree $t \in TL(A)$ and $p_x \in Pos(t, HX)$, let $QP^t_{HX}(p_x)$ denote the pair $(r^t_A(p_h), r^t_A(p_x))$ of states of A such that $p_x = p_h \cdot p$ and $\tilde{\lambda}_t(p_h) = H$. Let QP^t_{HX} denote the set $\{QP^t_{HX}(p_x) \mid p_x \in Pos(t, HX)\}$. Note that QP^t_{HX} is fixed over all $t \in TL(A)$ because of the f-path fixity of A. The fixed set is also denoted by QP^A_{HX}.

Now, we show some properties on f-path fixed TAs. For a tree $t \in \mathcal{T}_\Sigma$ and an FD $f = (H, X, Y)$, let $Pos^f_{Nblw}(t) = Pos(t) - \{p \cdot p' \mid p \in Pos(t, HX) \cup Pos(t, HY), p' \in Pos(t|_p) - \{\epsilon\}\}$.

Lemma 1: Suppose that a TA A is f-path fixed where $f = (H, X, Y)$, and $t_a \in TL(A)$ satisfies f. Then, there is some mapping $M_{t_a} : TL(A) \to TL(A)$ such that for any $t \in TL(A)$, $M_{t_a}(t)$ satisfies f, $Pos^f_{Nblw}(t) = Pos^f_{Nblw}(M_{t_a}(t))$, and for each $p \in Pos^f_{Nblw}(t)$, $r^t_A(p) = r^{M_{t_a}(t)}_A(p)$.

Proof. We construct three functions F_H, F_{HX}, F_{HY} from t_a, which are used to construct M_{t_a}. The first function F_H is a total function from $r^{t_a}_A(Pos(t_a, H))$ to $Pos(t_a, H)$ such that for each $q_h \in r^{t_a}_A(Pos(t_a, H))$, $r^{t_a}_A(F_H(q_h)) = q_h$. The second function F_{HX} is a total function from $QP^{t_a}_{HX}$ to $Pos(t_a, HX)$ such that for each $(q_h, q_x) \in QP^{t_a}_{HX}$, $F_H(q_h) \in Anc_{t_a}(F_{HX}((q_h, q_x)))$ and $r^{t_a}_A(F_{HX}((q_h, q_x))) = q_x$. The third function F_{HY} is a total function from $r^{t_a}_A(Pos(t_a, HY))$ to $Pos(t_a, HY)$ such that for each $q_y \in r^{t_a}_A(Pos(t_a, HY))$, $r^{t_a}_A(F_{HY}(q_y)) = q_y$.

Now, we construct $M_{t_a} : TL(A) \to TL(A)$ such that for each $t \in TL(A)$, $M_{t_a}(t)$ is the tree obtained from t by replacing $t|_{p_x}$ with $t_a|_{F_{HX}(QP^t_{HX}(p_x))}$ for each position $p_x \in Pos(t, HX)$, and $t|_{p_y}$ with $t_a|_{F_{HY}(r^t_A(p_y))}$ for each position $p_y \in Pos(t, HY)$. Note that this replacement is always possible because $QP^{t_a}_{HX} = QP^t_{HX}$ and $r^{t_a}_A(Pos(t_a, HY)) = r^t_A(Pos(t, HY))$ by the conditions (1) and (2) of f-path fixity. We see that $M_{t_a}(t) \in TL(A)$ for any tree $t \in TL(A)$ because for each position $p_z \in Pos(t, HX) \cup Pos(t, HY)$, $M_{t_a}(t)|_{p_z} \in TL(A, r^t_A(p_z))$. We also have that for each tree $t \in TL(A)$, $Pos^f_{Nblw}(t) = Pos^f_{Nblw}(M_{t_a}(t))$ and for each $p \in Pos^f_{Nblw}(t)$, $r^t_A(p) = r^{M_{t_a}(t)}_A(p)$. It is because $M_{t_a}(t)$ is obtained by changing only subtrees at positions in $Pos(t, HX) \cup Pos(t, HY)$, maintaining the states assigned to the positions.

Moreover, we can see that $M_{t_a}(t)$ satisfies f for each tree $t \in TL(A)$. Consider an arbitrary tree $t \in TL(A)$, and assume that $M_{t_a}(t)$ does not satisfy f. Then, there are two positions $p_{h1}, p_{h2} \in Pos(M_{t_a}(t), H)$ such that $X(M_{t_a}(t), p_{h1}) \cap X(M_{t_a}(t), p_{h2}) \neq \emptyset$ and $Y(M_{t_a}(t), p_{h1}) \cap Y(M_{t_a}(t), p_{h2}) = \emptyset$. Let $q_{h1} = r^{M_{t_a}(t)}_A(p_{h1})$ and $q_{h2} = r^{M_{t_a}(t)}_A(p_{h2})$. From the construction of F_{HX}, we have that for each $i \in \{1, 2\}$, $X(M_{t_a}(t), p_{hi}) \subseteq X(t_a, F_H(q_{hi}))$. Thus, $X(t_a, F_H(q_{h1})) \cap X(t_a, F_H(q_{h2})) \neq \emptyset$. On the other hand, from the construction of F_{HY} and the condition (3) of f-path fixity, we also have that $r^{M_{t_a}(t)}_A(Pos(M_{t_a}(t), p_{h1}, Y)) \cap r^{M_{t_a}(t)}_A(Pos(M_{t_a}(t), p_{h2}, Y)) = \emptyset$. For each $i \in \{1, 2\}$, from the condition (2) of f-path fixity and $r^{M_{t_a}(t)}_A(p_{hi}) = r^{t_a}_A(F_H(q_{hi}))$, $r^{M_{t_a}(t)}_A(Pos(M_{t_a}(t), p_{hi}, Y)) = r^{t_a}_A(Pos(t_a, F_H(q_{hi}), Y))$. Hence, $r^{t_a}_A(Pos(t_a, F_H(q_{h1}), Y)) \cap r^{t_a}_A(Pos(t_a, F_H(q_{h2}), Y)) = \emptyset$. Again, from the condition (3) of f-path fixity, $Y(t_a, F_H(q_{h1})) \cap Y(t_a, F_H(q_{h2})) = \emptyset$. Therefore, t_a does not satisfy f, and it is a contradiction. □

Next, we show that satisfiability of an FD $f = (H, X, Y)$ with respect to an f-path fixed TA A, i.e., whether $TL(A) \cap TL(f) \neq \emptyset$, is decidable. Now, for $q_x \in Q^A_{HX}$ let $k(q_x)$ denote the size of the maximum subset Q'_H of Q^A_H satisfying the following condition: For any two distinct states $q_{h1}, q_{h2} \in Q'_H$, $q_x \in Q^A_{q_{h1},X} \cap Q^A_{q_{h2},X}$ and $Q^A_{q_{h1},Y} \cap Q^A_{q_{h2},Y} = \emptyset$. Then, we have the following lemma.

Lemma 2: Suppose that A is an f-path fixed TA. Then, there is some tree $t \in TL(A)$ which satisfies f if and only if $|TL(A, q_x)| \geq k(q_x)$ for any $q_x \in Q^A_{HX}$.

Proof. Assume that $TL(A, q_x)$ contains at least $k(q_x)$ trees for any $q_x \in Q^A_{HX}$. We can consider the following two mappings M_{HX} and M_{HY}. The first mapping M_{HX} satisfies that $M_{HX}((q_h, q_x)) \in TL(A, q_x)$ for each $(q_h, q_x) \in QP^A_{HX}$, and for any $(q_{h1}, q_x), (q_{h2}, q_x) \in QP^A_{HX}$ such that $q_{h1} \neq q_{h2}$ and $Q^A_{q_{h1},Y} \cap Q^A_{q_{h2},Y} = \emptyset$, $M_{HX}((q_{h1}, q_x)) \neq M_{HX}((q_{h2}, q_x))$. Since $|TL(A, q_x)| \geq k(q_x)$ for each $q_x \in Q^A_{HX}$, M_{HX} is constructible. The second mapping M_{HY} satisfies that $M_{HY}(q_y) \in TL(A, q_y)$ for each $q_y \in Q^A_{HY}$. Now, there is some tree $t \in TL(A)$ such that $t|_{p_x} = M_{HX}(QP^t_{HX}(p_x))$ for each $p_x \in Pos(t, HX)$ and $t|_{p_y} = M_{HY}(r^t_A(p_y))$ for each $p_y \in Pos(t, HY)$. Then, for any two $p_{h1}, p_{h2} \in Pos(t, H)$, we have that $Y(t, p_{h1}) \cap Y(t, p_{h2}) = \emptyset \Rightarrow X(t, p_{h1}) \cap X(t, p_{h2}) = \emptyset$. Hence, t satisfies f.

Conversely, assume that $TL(A, q_x)$ contains at most $k(q_x)-1$ trees for some $q_x \in Q^A_{HX}$. Then, from the f-path fixity of A, for any $t \in TL(A)$, there are positions p_{h1} and p_{h2} in $Pos(t, H)$ such that (a) two distinct states $q_{h1}, q_{h2} \in Q^A_H$ are assigned to p_{h1} and p_{h2}, (b) $Q^A_{q_{h1},Y} \cap Q^A_{q_{h2},Y} = \emptyset$, and (c) for some positions $p_{x1} \in Pos(t, p_{h1}, X)$ and $p_{x2} \in Pos(t, p_{h2}, X)$, $r^t_A(p_{x1}) = r^t_A(p_{x2}) = q_x$ and $t|_{p_{x1}} = t|_{p_{x2}}$. Thus, we have that $X(t, p_{h1}) \cap X(t, p_{h2}) \neq \emptyset$ and $Y(t, p_{h1}) \cap Y(t, p_{h2}) = \emptyset$. Hence, there is no tree in $TL(A)$ satisfying f. □

For each $q_x \in Q^A_{HX}$, $k(q_x)$ is computable and it is decidable whether $|TL(A, q_x)| \geq k(q_x)$. Thus, we can decide whether there is some tree in $TL(A)$ satisfying f.

Lemma 3: Satisfiability of an FD $f = (H, X, Y)$ with respect to an f-path fixed TA A is decidable.

4.3 Details of the Decision Algorithm

We explain the details of our decision algorithm for ∞-secrecy with an FD. The detail of Step 1 is omitted because Step 1 is the same as Steps 1 and 2 of the algorithm proposed in our previous work [8].

4.3.1 Step 2: Division into f-Path Fixed TAs

In this step, the TA A_a obtained in Step 1 is divided into f-path fixed TAs. We assume that $A_a = (Q, \Sigma, Q^i, R)$ is a bottom-up deterministic TA without loss of generality. We give a procedure for dividing A_a into a finite set of f-path fixed TAs. First, we construct an unambiguous TA $A_f = (Q_f \cup \bar{Q}_f, \Sigma, Q^i_f, R_f)$ such that (a) $TL(A_f) = \mathcal{T}_\Sigma$, (b) $Q_f \cap \bar{Q}_f = \emptyset$, and (c) for any tree $t \in \mathcal{T}_\Sigma$, the run of A_f on t assigns states in Q_f to nodes on the path H, HX, or HY, and assigns states in $\bar{Q}_f$ to the other nodes v. The detail of A_f is presented in Appendix A. Then, we construct the intersection TA A_p of A_a and A_f constructed in the similar way as the product.

Here, we define the following binary relation $\prec$ over $Q \times Q_f$: $q' \prec q$ if there is some rule (q, a, e) in A_p such that q' appears in e. From the construction of A_f (see Appendix A.), no state in Q_f is recursive on A_f, and thus the reflective transitive closure $\prec^*$ of $\prec$ is a partial order. According to a topological ordering of $\prec^*$, starting from the smallest state, for each $q \in Q \times Q_f$, we do the equivalence transformation of A_p as follows: if the rule (q, a, e) exists in A_p, let Q_e be the set of states appearing in e. For each $Q' \subseteq Q_e$, create a new state $q_{Q'}$, and a new rule $(q_{Q'}, a, e')$ such that e' is the intersection RE of e and $({q'_1}^+ \& \cdots \& {q'_m}^+)$ where $Q' = \{q'_1, \ldots, q'_m\}$. Moreover, for every rule (q', a', e') such that q appears in e', substitute q in e' with $q_\emptyset \mid \cdots \mid q_{Q_e}$. The equivalence transformation eventually terminates because $\prec^*$ is a partial order. Let A_{dv} be the TA obtained by the equivalence transformation. By this transformation, for each state appearing on the label path HX or HY, the set of states assigned to their children position is fixed in any tree accepted by A_{dv}. Let Q^{ini}_{dv} be the set of the initial states of A_{dv}. For each initial state $\hat{q}_i \in Q^{ini}_{dv}$, the TA A^i_{dv} obtained by restricting the set of initial states to the singleton $\{\hat{q}_i\}$ is f-path fixed. We have that $TL(A_a) = TL(A_{dv}) = \bigcup_{\hat{q}_i \in Q^{ini}_{dv}} TL(A^i_{dv})$.

Lemma 4: Given a TA A_a and an FD f, A_a can be divided into a finite number of f-path fixed TAs $A_1, \ldots, A_n$ such that $TL(A_a) = \bigcup_i TL(A_i)$ and for every pair A_i and A_j $(1 \leq i, j \leq n, i \neq j)$, $TL(A_i) \cap TL(A_j) = \emptyset$.

Let $\mathcal{A}(A_a, f)$ denote the finite set of f-path fixed TAs with respect to A_a and f. Then, we have the following lemma on ∞-secrecy.

Lemma 5: Suppose that A_a is the TA obtained at Step 1 with respect to authorized queries $T_1, \ldots, T_n$ and their results $T_1(t), \ldots, T_n(t)$. Then, t is ∞-secret if and only if there is some $A_i \in \mathcal{A}(A_a, f)$ such that $C(A_i, T_s, f) = \{T_s(t) \mid t \in TL(A_i) \cap TL(f)\}$ is infinite.

Proof. For the if part, because $C(A_i, T_s, f) \subseteq C(A_a, T_s, f)$, if $C(A_i, T_s, f)$ is infinite, then $C(A_a, T_s, f)$ is infinite. Conversely, assume that $C(A_a, T_s, f)$ is infinite. Since $\mathcal{A}(A_a, f)$ is finite and $C(A_a, T_s, f) = \bigcup_{A_i \in \mathcal{A}(A_a, f)} C(A_i, T_s, f)$, there is a TA $A_i \in \mathcal{A}(A_a, f)$ such that $C(A_i, T_s, f)$ is infinite. □

4.3.2 Step 3: Decision of ∞-Secrecy

In this step, we decide whether t is ∞-secret using the necessary and sufficient condition stated in Lemma 5. To show the decidability of the condition, we just show that it is decidable whether $C(A_i, T_s, f)$ is infinite for one f-path fixed TA A_i because $\mathcal{A}(A_a, f)$ is finite. Now, we give a necessary and sufficient condition for $C(A_i, T_s, f)$ to be infinite.

Lemma 6: Suppose that A is an f-path fixed TA and that T_s is a composition of a TRTT T^R_s and a DTT T^D. Then, $TL(A) \cap TL(f) \neq \emptyset$ and $T_s(TL(A))$ is infinite if and only if $C(A, T_s, f)$ is infinite.

Proof. The if part is obvious because $T_s(TL(A)) \supseteq C(A, T_s, f)$, and $C(A, T_s, f) \neq \emptyset$ implies $TL(A) \cap TL(f) \neq \emptyset$. So, we give the proof for the only if part.

Before starting the proof, we give a definition for a pair

tree of input and output trees on a TRTT. Consider an arbitrary tree t and the output tree $T_s^R(t)$ of the TRTT T_s^R on t. Since a TRTT can only relabel nodes, t and $T_s^R(t)$ have the same set of the positions, i.e., $Pos(t) = Pos(T_s^R(t))$. This enables us to define a pair tree, denoted by $[t, T_s^R(t)]$, of t and $T_s^R(t)$ such that $Pos([t, T_s^R(t)]) = Pos(t)$ and for each position $p \in Pos([t, T_s^R(t)])$, $\lambda_{[t,T_s^R(t)]}(p) = (\lambda_t(p), \lambda_{T_s^R(t)}(p))$. For example, when $t = a(bc)$ and $T_s^R(t) = x(yz)$, $[t, T_s^R(t)] = (a, x)((b, y)(c, z))$. Let π_i denote the i-th projection of the label pair (a, x), i.e., $\pi_1((a, x)) = a$ and $\pi_2((a, x)) = x$. We extend the projection to a label path $\ell = l_1 \cdots l_n$, i.e., $\pi_i(\ell) = \pi_i(l_1) \cdots \pi_i(l_n)$. Given an f-path fixed TA A and a TRTT T_s^R where $f = (H, X, Y)$, we can construct a TA A_{io} such that $TL(A_{io}) = \{[t, T_s^R(t)] \mid t \in TL(A)\}$ by using almost the same technique as type inference. Here, consider the first element projection $\pi_1(A_{io})$ of A_{io}, which is obtained by replacing each rule $(\bar{q}, (a, \alpha), \bar{e})$ of A_{io} with $(\bar{q}, a, \bar{e})$. Then, $\pi_1(A_{io})$ is f-path fixed because A is f-path fixed and a TRTT cannot distinguish nodes reachable via the same path from the root. Thus, from Lemma 1, if there is some tree $[t_a, T_s^R(t_a)]$ such that t_a satisfies f, there is some mapping $M_{t_a} : TL(\pi_1(A_{io})) \to TL(\pi_1(A_{io}))$ such that for each tree $t \in TL(\pi_1(A_{io}))$, $M_{t_a}(t)$ satisfies f, $Pos^f_{Nblw}(t) = Pos^f_{Nblw}(M_{t_a}(t))$, and for each $p \in Pos^f_{Nblw}(t)$, $r^t_{\pi_1(A_{io})}(p) = r^{M_{t_a}(t)}_{\pi_1(A_{io})}(p)$. We extend M_{t_a} to a pair tree in $TL(A_{io})$, i.e., $M_{t_a}([t, T_s^R(t)]) = [M_{t_a}(t), T_s^R(M_{t_a}(t))]$.

Assume that $TL(A) \cap TL(f) \neq \emptyset$ and $T_s(TL(A))$ is infinite. Since $TL(A) \cap TL(f) \neq \emptyset$, there is some tree $t_a \in TL(A)$ satisfying f, and $\bar{t}_a = [t_a, T_s^R(t_a)] \in TL(A_{io})$. Thus, there is some mapping M_{t_a} as stated above. On the other hand, since $T_s(TL(A))$ is infinite, there is some state q_c of A_{io} and some tree $\bar{t}' = [t', T_s^R(t')]$ such that:

1. for some position p_c of $\bar{t}'$, the symbol \$ does not appear in $\pi_2(\tilde{\lambda}_{\bar{t}'}(p_c))$, $r^{\bar{t}'}_{A_{io}}(p_c) = q_c$, and the state q_c appears at least twice in $\tilde{r}^{\bar{t}'}_{A_{io}}(p_c)$; or
2. for some position $p_c = p' \cdot i$ of $\bar{t}'$ where $i \in \mathbb{N}$, the symbol \$ does not appear in $\pi_2(\tilde{\lambda}_{\bar{t}'}(p_c))$, $r^{\bar{t}'}_{A_{io}}(p_c) = q_c$, $r^{\bar{t}'}_{A_{io}}(p') = q'$, and for the rule (q', a, e) and some words α, β, γ such that β includes q_c, the word $\alpha\beta^j\gamma$ is in $L(e)$ for any positive integer $j \geq 1$.

In other words, $\bar{t}'$ is pumpable at p_c vertically and/or horizontally. In addition, we can see that for any two distinct trees $[t'', T_s^R(t'')]$ and $[t''', T_s^R(t''')]$ obtained by pumping $\bar{t}'$ at p_c, $t'' \neq t'''$ and also $T_s(t'') \neq T_s(t''')$. Here, we have three cases according to where the pumpable position p_c is with respect to f. We shall show, in each case, that there are an infinite number of trees in $TL(A)$ satisfying f such that the outputs of T_s on them are all distinct.

(i) (See Fig. 6 (i)) Assume that neither HX nor HY is a prefix of $\tilde{\lambda}_{t'}(p_c)$. Let $\bar{t}'' = [t'', T_s^R(t'')] = M_{t_a}(\bar{t}')$. Then, t'' satisfies f by Lemma 1. Moreover, M_{t_a} has no influence on the position p_c, and thus $p_c \in Pos(\bar{t}'')$ and $r^{\bar{t}''}_{A_{io}}(p_c) = q_c$. By pumping $\bar{t}''$ at p_c on q_c, we can obtain an infinite number of trees such that they satisfy f and the outputs of T_s on them are all distinct.

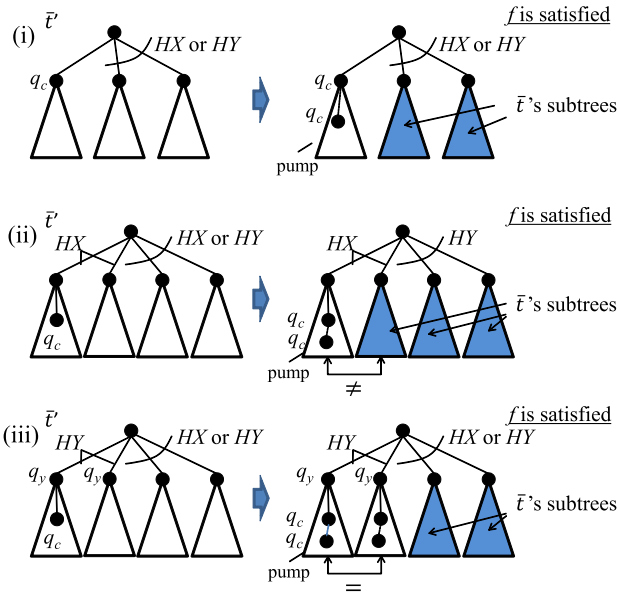


Fig. 6 The proof of Lemma 6.

(ii) (See Fig. 6 (ii)) Assume that HX is a prefix of $\tilde{\lambda}_{t'}(p_c)$. Let p_x and p_h be the ancestor positions of p_c such that $\tilde{\lambda}_{t'}(p_x) = HX$ and $\tilde{\lambda}_{t'}(p_h) = H$ respectively. Let $\bar{t}'' = [t'', T_s^R(t'')] = M_{t_a}(\bar{t}')$, and let $\bar{t}_e = [t_e, T_s^R(t_e)]$ be the tree such that $\bar{t}_e = \bar{t}'[p_x \leftarrow \bar{t}'|_{p_x}]$. Then, since $\bar{t}_e|_{p_x} = \bar{t}'|_{p_x}$, we have that $p_c \in Pos(\bar{t}_e)$ and $r^{\bar{t}_e}_{A_{io}}(p_c) = q_c$. Note that t'' satisfies f but t_e might not satisfy f. Since the difference of t'' and t_e is only the subtree at p_x, for any two positions p_{h1} and p_{h2} in $Pos(t_e, H) - \{p_h\}$, $X(t_e, p_{h1}) \cap X(t_e, p_{h2}) \neq \emptyset$ implies $Y(t_e, p_{h1}) \cap Y(t_e, p_{h2}) \neq \emptyset$. We also have that $Y(t'', p_h) = Y(t_e, p_h)$. The difference of t'' and t_e affects only the dependencies between $t_e|_{p_h}$ and the other subtrees in $H(t_e)$. Now, note that $TL(A_{io}, r^{\bar{t}_e}_{A_{io}}(p_x))$ is infinite because $p_x \in Anc_{\bar{t}_e}(p_c)$ and $TL(A_{io}, r^{\bar{t}_e}_{A_{io}}(p_c))$ is infinite. So, we can consider a tree $\bar{t}_x = [t_x, T_s^R(t_x)] \in TL(A_{io}, r^{\bar{t}_e}_{A_{io}}(p_x))$ such that $t_x \neq t''_x$ for any $t''_x \in HX(t'')$. Let $\bar{t}'_e = [t'_e, T_s^R(t'_e)]$ be the tree such that $\bar{t}'_e = \bar{t}''[p_x \leftarrow \bar{t}_x]$. Then, t'_e satisfies f. By pumping $\bar{t}_e$ at p_c on q_c, an infinite number of distinct subtrees can be derived at p_c without changing the other part. Hence, there are an infinite number of distinct trees $[t'_e, T_s^R(t'_e)]$ such that $t'_e|_{p_x} \neq t'_x$ for any $t'_x \in HX(t'_e)$. Therefore, we can obtain an infinite number of trees such that they satisfy f and the outputs of T_s on them are all distinct.

(iii) (See Fig. 6 (iii)) Assume that HY is a prefix of $\tilde{\lambda}_{t'}(p_c)$. Let p_y and p_h be the ancestor positions of p_c such that $\tilde{\lambda}_{t'}(p_y) = HY$ and $\tilde{\lambda}_{t'}(p_h) = H$ respectively. Let $q_y = r^{\bar{t}'}_{A_{io}}(p_y)$. Let $\bar{t}'' = [t'', T_s^R(t'')] = M_{t_a}(\bar{t}')$, and let $\bar{t}_e = [t_e, T_s^R(t_e)]$ be the tree obtained from $\bar{t}''$ by replacing each subtree at each position p such that $r^{\bar{t}''}_{A_{io}}(p) = r^{\bar{t}''}_{A_{io}}(p_y)$ and $\tilde{\lambda}_{t''}(p) = HY$ with $\bar{t}'|_{p_y}$. Then, since $\bar{t}_e|_{p_y} = \bar{t}'|_{p_y}$, we have that $p_c \in Pos(\bar{t}_e)$ and

$r^{\bar{t}_e}_{A_{io}}(p_c) = q_c$. We show that t_e satisfies f. Let $Pos^{rp}_y = \{p \mid \tilde{\lambda}_{t_e}(p) = HY, r^{\bar{t}_e}_{A_{io}}(p) = r^{\bar{t}_e}_{A_{io}}(p_y)\}$ and $Pos^{rp}_h = \{p \mid p' \in Pos^{rp}_y, p \in Anc_{t_e}(p'), \tilde{\lambda}_{t_e}(p) = H\}$. Then, for any two $p_1, p_2 \in Pos^{rp}_h$, $t'|_{p_y} \in Y(t_e, p_1) \cap Y(t_e, p_2)$. We also have that for any q_{y1}, q_{y2} in $Q^{\pi_1(A_{io})}_{HY}$, $q_{y1} \neq q_{y2}$ implies $TL(A_{io}, q_{y1}) \cap TL(A_{io}, q_{y2}) = \emptyset$ by the f-path fixity. It holds that $Y(t'', p_1) \cap Y(t'', p_2) \neq \emptyset$ implies $Y(t_e, p_1) \cap Y(t_e, p_2) \neq \emptyset$ for any two $p_1, p_2 \in Pos(t'', H) = Pos(t_e, H)$. Thus, t_e satisfies f. By pumping $\bar{t}_e$ at p_c on q_c, an infinite number of distinct subtrees can be derived at p_c without changing the other part. The same subtrees can be derived at p in $Pos(t_e, HY)$ such that $r^{\bar{t}_e}_{A_{io}}(p) = q_c$. Therefore, we can obtain an infinite number of trees such that they satisfy f and the outputs of T_s on them are all distinct. □

From Lemma 3, It is decidable whether $TL(A) \cap TL(f) \neq \emptyset$. It is also decidable whether $T_s(TL(A))$ is infinite because a TA which recognizes $T_s(TL(A))$ can be constructed from T_s and A by type inference, and the finiteness of a regular tree language is decidable. Thus, from Lemmas 4 and 5, we have proved the correctness of our algorithm.

Theorem 1: ∞-SIAF is decidable when the unauthorized query is restricted to a composition of TRTTs and a DTT.

5. Discussion

We have provided a decision algorithm for ∞-SIAF when the unauthorized query is restricted to a composition of TRTTs and a DTT. When the unauthorized query includes BRTTs, our algorithm does not work well because Lemma 6 does not hold. For example, consider a TA A, an FD f, and T_s such that:

- The rule set of A contains

$$(S, s, U), (U, u, A_1A_2), (A_1, a, H), (A_2, a, HB),$$
$$(B, b, \epsilon), (H, h, XY), (X, x, \epsilon), (Y, y, C^*), (C, c, \epsilon),$$

- $f = (suah, x, y)$,
- $T_s = T^D \circ T_3 \circ T_2 \circ T_1$
  - T_1: relabels the node assigned A_2 to with a_2.
  - T_2: relabels the node which is a descendant of the node labeled with a_2 and the label is y with y_2.
  - T_3: if some node exists below the node labeled with x or y, relabels the u-labeled node with \$.
  - T^D: the topdown deleting tree transducer.

Then, A is an f-path fixed TA. Note that T_1 and T_3 can be represented by BRTTs but not by TRTTs. $TL(A)$ contains an infinite number of trees satisfying f, and $T_s(TL(A))$ is infinite. However, $\{T_s(t) \mid t \in TL(A) \cap TL(f)\}$ contains only two trees, s and $s(u(a(h(x, y)), a_2(h(x, y_2), b)))$. In this case, the nodes reachable from the root through the path *suahy* are distinguished by $T_2 \circ T_1$. Our decision algorithm requires that T_s assigns the same states to the nodes reachable through the same path. If T_s includes BRTTs, the condition is not always satisfied. In such a case, our algorithm does not work well. Decidability of ∞-SIAF for more powerful query models is an open problem.

6. Conclusion

This paper has discussed verification of the security against inference attacks on XML databases, considering a functional dependency. We have provided a decision algorithm for ∞-secrecy in the presence of a simple functional dependency when the unauthorized query is represented by a deterministic topdown tree transducer.

One of our future work is to investigate a more powerful class of unauthorized queries for which ∞-secrecy problem with a functional dependency is decidable. As another future work, we would like to extend the verification method to be able to deal with multiple FDs or more complicated FDs. Moreover, we would like to propose an algorithm to solve the k-secrecy problem with FDs for any positive integer $k > 1$.

References

[1] D.E.R. Denning, Cryptography and Data Security, Addison-Wesley, 1982.
[2] M. Morgenstern, "Security and inference in multilevel database and knowledge-base systems," Proc. 1987 ACM SIGMOD International Conference on Management of Data, pp.357–373, 1987.
[3] I. Moskowitz and L. Chang, "An entropy-based framework for database inference," Proc. 3rd International Workshop on Information Hiding, LNCS 1768, pp.405–418, 1999.
[4] K. Zhang, "IRI: A quantitative approach to inference analysis in relational databases," Database Security XI, pp.279–290, 1998.
[5] A. Deutsch and Y. Papakonstantinou, "Privacy in database publishing," Proc. 10th International Conference on Database Theory, pp.230–245, 2005.
[6] G. Miklau and D. Suciu, "A formal analysis of information disclosure in data exchange," J. Computer and System Sciences, vol.73, no.3, pp.507–534, 2007.
[7] A. Machanavajjhala, J. Gehrke, D. Kifer, and M. Venkitasubramaniam, "ℓ-diversity: Privacy beyond k-anonymity," Proc. 22nd International Conference on Data Engineering, p.24, 2006.
[8] K. Hashimoto, K. Sakano, F. Takasuka, Y. Ishihara, and T. Fujiwara, "Verification of the security against inference attacks on XML databases," IEICE Trans. Inf. & Syst., vol.E92-D, no.5, pp.1022–1032, May 2009.
[9] X. Yang and C. Li, "Secure XML publishing without information leakage in the presence of data inference," Proc. 30th International Conference on Very Large Data Bases, pp.96–107, 2004.
[10] D. Suciu, "The XML typechecking problem," SIGMOD Record, vol.31, no.1, pp.89–96, 2002.
[11] W. Martens and F. Neven, "On the complexity of typechecking topdown XML transformations," Theor. Comput. Sci., vol.336, no.1, pp.153–180, 2005.
[12] G. Miklau and D. Suciu, "Containment and equivalence for a fragment of XPath," J. ACM, vol.51, no.1, pp.2–45, 2004.
[13] S. Maneth, A. Berlea, T. Perst, and H. Seidl, "XML type checking with macro tree transducers," Proc. 24th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, pp.283–294, 2005.
[14] J. Clark and S. DeRose, "XML path language (XPath) version 1.0." Available on: http://www.w3.org/TR/xpath, 1999.

[15] M. Arenas and L. Libkin, "A normal form for XML documents," ACM Trans. Database Syst., vol.29, pp.195–232, 2004.
[16] C. Yu and H.V. Jagadish, "XML schema refinement through redundancy detection and normalization," The VLDB Journal, vol.17, pp.203–223, 2008.

Appendix A: An FD Automaton

Here, we give the FD automaton A_f (see Sect. 4) with respect to $f = (H, X, Y)$. First, we construct two TAs A_{HX} and A_{HY} as follows. Let $H = h_1 \cdots h_n$ and $X = x_1 \cdots x_m$. Then, $A_{HX} = (Q_X^H \cup \bar{Q}_X^H, \Sigma, \{q_{h_1}, \bar{q}_{h_1}, q_{\bar{h}_1}\}, R_X^H)$ where

$$\begin{aligned} Q_X^H &= \{q_{h_1}, \ldots, q_{h_n}\} \cup \{q_{x_1}, \ldots, q_{x_m}\}, \\ \bar{Q}_X^H &= \{\bar{q}_{h_1}, \ldots, \bar{q}_{h_{n-1}}\} \cup \{q_{\bar{h}_1}, \ldots, q_{\bar{h}_n}\} \\ &\quad \cup \{\bar{q}_{x_1}, \ldots, \bar{q}_{x_{m-1}}\} \cup \{q_{\bar{x}_1}, \ldots, q_{\bar{x}_m}\} \cup \{q_{any}\}, \end{aligned}$$

and R_X^H consists of the following transition rules:

- for each i $(1 \le i \le n-2)$,
  - $(q_{h_i}, h_i, ({q_{h_{i+1}}}^+ \,\&\, {\bar{q}_{h_{i+1}}}^* \,\&\, {q_{\bar{h}_{i+1}}}^*))$,
  - $(\bar{q}_{h_i}, h_i, ({\bar{q}_{h_{i+1}}}^* \,\&\, {q_{\bar{h}_{i+1}}}^*))$,
  - $(q_{\bar{h}_i}, \alpha, {q_{any}}^*)$ for each $\alpha \in \Sigma - \{h_i\}$,
- for $i = n-1$,
  - $(q_{h_i}, h_i, ({q_{h_{i+1}}}^+ \,\&\, {q_{\bar{h}_{i+1}}}^*))$,
  - $(\bar{q}_{h_i}, h_i, {q_{\bar{h}_{i+1}}}^*)$,
  - $(q_{\bar{h}_i}, \alpha, {q_{any}}^*)$ for each $\alpha \in \Sigma - \{h_i\}$,
- for $i = n$,
  - $(q_{h_i}, h_i, ({q_{x_1}}^* \,\&\, {\bar{q}_{x_1}}^* \,\&\, {q_{\bar{x}_1}}^*))$,
  - $(q_{\bar{h}_i}, \alpha, {q_{any}}^*)$ for each $\alpha \in \Sigma - \{h_i\}$,
- for each j $(1 \le j \le m-2)$,
  - $(q_{x_j}, x_j, ({q_{x_{j+1}}}^+ \,\&\, {\bar{q}_{x_{j+1}}}^* \,\&\, {q_{\bar{x}_{j+1}}}^*))$,
  - $(\bar{q}_{x_j}, x_j, ({\bar{q}_{x_{j+1}}}^* \,\&\, {q_{\bar{x}_{j+1}}}^*))$,
  - $(q_{\bar{x}_j}, \alpha, {q_{any}}^*)$ for each $\alpha \in \Sigma - \{x_j\}$,
- for $j = m-1$,
  - $(q_{x_j}, x_j, ({q_{x_{j+1}}}^+ \,\&\, {q_{\bar{x}_{j+1}}}^*))$,
  - $(\bar{q}_{x_j}, x_j, {q_{\bar{x}_{j+1}}}^*)$,
  - $(q_{\bar{x}_j}, \alpha, {q_{any}}^*)$ for each $\alpha \in \Sigma - \{x_j\}$,
- for $j = m$,
  - $(q_{x_j}, x_j, {q_{any}}^*)$,
  - $(q_{\bar{x}_j}, \alpha, {q_{any}}^*)$ for each $\alpha \in \Sigma - \{x_j\}$, and
- $(q_{any}, \alpha, {q_{any}}^*)$ for each $\alpha \in \Sigma$.

We construct A_{HY} in the same way. Then, $A_f = (Q_f \cup \bar{Q}_f, \Sigma, Q^i, R_f)$ is the intersection TA of A_{HX} and A_{HY} where $Q_f = Q_X^H \times (Q_Y^H \cup \bar{Q}_Y^H) \cup (Q_X^H \cup \bar{Q}_X^H) \times Q_Y^H$ and $\bar{Q}_f = \bar{Q}_X^H \times \bar{Q}_Y^H$. Since A_{HX} and A_{HY} are unambiguous, so is A_f. In addition, for any tree $t \in \mathcal{T}_\Sigma$ and position $p \in Pos(t)$, if there is some position p' such that $p \in Anc_t(p')$ and $\tilde{\lambda}_t(p')$ is HX, HY, or H, then $r^t_{A_f}(p) \in Q_f$; otherwise, $r^t_{A_f}(p) \in \bar{Q}_f$.

Kenji Hashimoto received the B.E., M.E., and Ph.D. degrees in information and computer sciences from Osaka University in 2004, 2006, and 2009, respectively. He is currently an Assistant Professor of Graduate School of Information Science, Nara Institute of Science and Technology. His research interests include formal language, database theory, and information security.

Hiroto Kawai received the B.E. and M.E. degrees in information and computer sciences from Osaka University in 2009 and 2011, respectively. He is currently with NTT WEST Corporation.

Yasunori Ishihara received the B.E., M.E., and Ph.D. degrees in information and computer sciences from Osaka University in 1990, 1992, and 1995, respectively. In 1994, he joined the faculty of Graduate School of Information Science, Nara Institute of Science and Technology. From 1999 to 2002, he was an Assistant Professor of Department of Informatics and Mathematical Science, Osaka University. Since April 2002, he has been an Associate Professor of Graduate School of Information Science and Technology, Osaka University. His research interests include database theory and information security.

Toru Fujiwara received the B.E., M.E., and Ph.D. degrees in information and computer sciences form Osaka University in 1981, 1983, and 1986, respectively. In 1986, he joined the faculty of Osaka University. Since 1997, he has been a Professor at Osaka University. He is currently with Graduate School of Information Science and Technology. In 1998–2000, he was a Professor at Graduate School of Information Science, Nara Institute of Science and Technology, simultaneously. His current research interests include coding theory and information security.