Toward Security Verification against Inference Attacks on Data Trees

Ryo Iwase, Yasunori Ishihara, Toru Fujiwara

Graduate School of Information Science and Technology, Osaka University, Japan
Inference attacks

- Attacker infers the sensitive data from
  - results of authorized queries
  - prior knowledge about the database

![Diagram of database, attacker, and sensitive data]
Example of an inference attack

• An XML document $I$

```
faculty
  student {$400}
  name {Alice} origin {USA}
  student {$500}
  name {Bob} origin {Japan}
  student {$800}
  name {Carol} origin {USA}
  student {$450}
  name {Dave} origin {Vietnam}
```

• The unauthorized queries $T_A, T_B, T_C$, and $T_D$
  – extract the amount of the scholarship of Alice, Bob, Carol, and Dave, respectively
\( T_1 \): extracting the name and the origin of each student

\[ \text{name-origin} \]

- student \{-1\}
  - name \{Alice\} origin \{USA\}
  - student \{-1\}
    - name \{Bob\} origin \{Japan\}
  - student \{-1\}
    - name \{Carol\} origin \{USA\}
  - student \{-1\}
    - name \{Dave\} origin \{Vietnam\}

\( T_2 \): extracting the origin of the student who receives the most amount of scholarship

\[ \text{origin-scholarship} \]

- student \{-1\}
  - origin \{USA\}

\( T_3 \): extracting the origin and scholarship of the student who receives the second most amount of scholarship

\[ \text{name-scholarship} \]

- student \{\$500\}
  - name \{Bob\}

DB schema

- faculty
  - student \{@scholarship\}
    - name \{@\} origin \{@\}
  - student \{@scholarship\}
    - name \{@\} origin \{@\}
  - student \{@\}
    - name \{@\} origin \{@\}
$T_1$: extracting the name and the origin of each student

$T_2$: extracting the origin of the student who receives the most amount of scholarship

$T_3$: extracting the origin and scholarship of the student who receives the second most amount of scholarship
Previous work

- A verification method against inference attacks **not considering data values** [1]
  - Information available to a user
    - DB schema
    - Authorized queries and the results of them
    - Unauthorized query extracting the sensitive information
  - A database is **infinitely secret**
    - if an attacker cannot narrow down the candidate for sensitive information to finite by available information

Definition of infinite secrecy

D is infinitely secret w.r.t. $T_S$ if $|C| = \infty$

$D$ is an instance of $A_G$,

$T_1(D') = T_1(D), \ldots, T_n(D') = T_n(D)$

$U = \{D' \mid D' \text{ is an instance of } A_G, \quad T_1(D') = T_1(D), \ldots, T_n(D') = T_n(D)\}$

$C = \{T_S(D') \mid D' \in U\}$

candidates for the sensitive data

database $D$

candidates for the database

unauthorized query $T_S$

results of authorized queries $T_1(D), \ldots, T_n(D)$

authorized queries $T_1, \ldots, T_n$

attacker

schema $A_G$
Purpose and requirements

• **Purpose**
  – To provide a verification method for infinite secrecy on data trees
    • Each node can have a data value

• **Requirements**
  1. Practically significant queries are allowed
  2. Input/output data trees of queries can be represented correctly
     • The construction of the set of input/output trees of queries from the set of output/input trees need to be possible
  3. Infiniteness of the set of candidates for sensitive data is decidable
Main result

• To introduce tree transducers on data trees
  – Operations correspond to projection, selection, and natural join in the relational algebra

• To provide data tree types
  – in order to represent input/output data trees of queries correctly

• To provide an algorithm for deciding infiniteness of data tree types
Contents

• Model of XML databases
• Decidability of infinite secrecy
• Conclusion and future work
Model of XML databases

• XML document:
  – data tree
    • Each node has a label and a data value
    • Data domain can be an infinite set

• XML schema:
  – non-deterministic finite tree automaton

• Query:
  – composition of seven kinds of tree transducers
    • Three : existing [1] --- independent of data values
    • Four : new --- dependent on data values

Existing tree transducers

- **Top-down relabeling tree transducer**
- **Bottom-up relabeling tree transducer**
- **Deleting tree transducer**

Relabel nodes according to the state and label of the parent.

Relabel nodes according to their own labels and the states of their children.

Delete nodes labeled by # and subtrees rooted by $.

$q_1$, $q_2$, $q_3$, $q_{1a}$ are special symbols.
New tree transducers

- Correspond to projection:
  - Rewrite data values of nodes labeled by “a” to “10”

- Correspond to selection:
  - Relabel nodes which have a label “a” and a data value “2” to “x”

- Used for representing natural join:
  - Relabel nodes which have the minimum value of the nodes labeled by “a” to “x”

- Relabel nodes which have the maximum value of the nodes labeled by “a” to “x”
Model of queries

• Restriction
  – No constituent tree transducers of the unauthorized query relabel any nodes to #
(1) choose a pair of nodes

(2) relabel their nodes to a new label by a bottom-up/top-down tree transducer

(3) Relabel their nodes to a new label by a min-data-relabeling/max-data relabeling tree transducer

(4) delete nodes by deleting tree transducer

If two nodes have the same value, they are relabeled to the same label.
Data tree types

• are used for representing the set of input/output data trees of tree transducers
• Two kinds of variables are used for representing data value of nodes
• 3-tuple $\langle A, \theta, E \rangle$
  – $A$ is a non-deterministic tree automaton
  – $\theta$ is a mapping : (state of $A$, label) $\mapsto$ variable
  – $E$ is a finite set of conditional expressions on data value of nodes
S-variable $\bar{x}$: all data values of nodes which have the same s-variable must be the same

$\theta(q_1, \text{student}) = \bar{x}$
$\theta(q_2, \text{student}) = \tilde{y}$

$E = \{\bar{x} < 300, 400 < \tilde{y} < 850\}$

M-variable $\tilde{y}$: all data values of nodes which have the same m-variable are not necessarily the same

- $D$ is valid against $A$
- Data values satisfy all conditional expressions in $E$

$D$ is in $\langle A, \theta, E \rangle$
How to decide infinite secrecy

(1) compute $U$ by inverse type inference on the authorized queries

$candidates for the database$

$U = \{D' \mid D' \text{ is an instance of } A_G, \quad T_1(D') = T_1(D), \ldots, T_n(D') = T_n(D)\}$

$candidates for the sensitive data$

$C = \{T_S(D') \mid D' \in U\}$

$D$ is infinitely secret w.r.t. $T_S$ if $|C| = \infty$

(2) compute $C$ by type inference on the unauthorized query

$\text{(3) Decide infiniteness of } C$

$\text{results of authorized queries}$

$T_1(D), \ldots, T_n(D)$

$\text{attacker}$

$\text{authorized queries}$

$T_1, \ldots, T_n$
Example of (inverse) type inference

Automaton $A' = A$

$(q_0, \text{faculty}, q_1^*)$
$(q_1, \text{student}, q_2 q_3)$
$(q_2, \text{name}, \varepsilon)$
$(q_3, \text{origin}, \varepsilon)$

$\theta'(q_1, \text{student}) = \tilde{x}'$
$\theta'(q_2, \text{name}) = y$
$\theta'(q_3, \text{origin}) = z$

$E' = E \cup \{x = d\}$

$\langle A', \theta', E' \rangle$

data-rewriting tree transducer

Automaton $A$

$(q_0, \text{faculty}, q_1^*)$
$(q_1, \text{student}, q_2 q_3)$
$(q_2, \text{name}, \varepsilon)$
$(q_3, \text{origin}, \varepsilon)$

$\theta(q_1, \text{student}) = x$
$\theta(q_2, \text{name}) = y$
$\theta(q_3, \text{origin}) = z$

$E = \{y = d_1, z \neq d_2\}$

$\langle A, \theta, E \rangle$
Algorithm for deciding infiniteness of data tree types

(1) Check whether the number of trees accepted by $A$ is infinite

A data tree type $\langle A, \theta, E \rangle$

(2) Check whether there exists a variable which can have infinite number of values

output “infinite”

output “finite”

output “finite”

output “finite”
How to check the existence of a variable with infinite number of values

\[ E = \{ 1 \in \bar{x}_1, 2 \not\in \bar{x}_2, 3 \in \bar{x}_1, \bar{x}_1 \subseteq \bar{x}_2 \} \]
How to check the existence of a variable with infinite number of values

\[ E = \{1 \in \bar{x}_1, \; 2 \notin \bar{x}_2, \; 3 \in \bar{x}_1, \; \bar{x}_1 \subseteq \bar{x}_2\} \]
How to check the existence of a variable with infinite number of values

\[ E = \{1 \in \tilde{x}_1, 2 \notin \tilde{x}_2, 3 \in \tilde{x}_1, \tilde{x}_1 \subseteq \tilde{x}_2\} \]
How to check the existence of a variable with infinite number of values

If all conditional expressions in $E$ are satisfied and a zone with infinite number of data values is assigned to at least one variable, there exists a variable with infinite number of values.

Assign zones to each $m$-variable

$$E = \{1 \in x_1, 2 \not\in x_2, 3 \in x_1, x_1 \subseteq x_2\}$$
Conclusions

• A verification method against inference attacks on data trees
  – Tree transducers on data trees
  – Type inference and inverse type inference on tree transducers
  – Algorithm for deciding infiniteness of data tree types
Ongoing and future work

• To prove that (inverse) type inference are possible on all tree transducers

<table>
<thead>
<tr>
<th></th>
<th>data-rewriting</th>
<th>data-relabeling</th>
<th>min-data-relabeling</th>
<th>max-data-relabeling</th>
</tr>
</thead>
<tbody>
<tr>
<td>inverse type inference</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>type inference</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

• To evaluate the complexity of our method

• To consider inference attacks using functional dependencies on data trees