

Theorem-proving anonymity of infinite-state systems

Yoshinobu Kawabe^{*}, Ken Mano, Hideki Sakurada, Yasuyuki Tsukada

NTT Communication Science Laboratories, NTT Corporation, Japan

Received 30 May 2005; received in revised form 8 February 2006; accepted 26 June 2006

Available online 23 August 2006

Communicated by D. Basin

Keywords: Anonymity; Verification; Infinite-state system; I/O-automaton; Theorem proving; Formal methods; Safety/security in digital systems

1. Introduction

The notion of anonymity is present in many fields of human activity, e.g. anonymous donation, voting, submitting poems (anonymously), whistle-blowing, and reviewing technical papers. On the Internet, there are also many services and protocols where anonymity should be provided. For example, an electronic voting system should guarantee anonymity to prevent the disclosure of who voted for which candidate.

Recently, there have been several studies based on formal methods that analyzed the anonymity of distributed systems [1–3]. A computer-assisted proof technique with model-checking also appeared in [5], but this technique cannot handle infinite-state systems directly. A proof technique that incorporates theorem-proving makes it possible to handle the anonymity of infinite-state systems. This paper presents an inductive method for verifying the anonymity of distributed systems with a theorem prover. We employ an I/O-automaton [4] to describe a distributed, possibly infinite-state, system. We first extend the formulation of anonymity described in [5] to devise the concept of *trace anonymity*, which is defined with the set of traces of an I/O-automaton. Then,

we introduce a proof technique with an *anonymous simulation*, which is an inductive method for proving trace anonymity. We show the existence of an anonymous simulation implies trace anonymity. We also demonstrate theorem-proving anonymity for an infinite-state system.

2. I/O-automaton

I/O-automaton X has a set of actions $\text{sig}(X)$, a set of states $\text{states}(X)$, a set of initial states $\text{start}(X) \subset \text{states}(X)$ and a set of transitions $\text{trans}(X) \subset \text{states}(X) \times \text{sig}(X) \times \text{states}(X)$. We use $\text{in}(X)$, $\text{out}(X)$ and $\text{int}(X)$ for the set of input, output and internal actions, respectively; that is, $\text{sig}(X) = \text{in}(X) \cup \text{out}(X) \cup \text{int}(X)$. We assume that $\text{in}(X)$, $\text{out}(X)$ and $\text{int}(X)$ are disjoint. We define $\text{ext}(X) = \text{out}(X) \cup \text{in}(X)$ whose element is called an external action. For simplicity, we only deal with I/O-automaton X satisfying $\text{in}(X) = \emptyset$; that is, we assume that $\text{ext}(X) = \text{out}(X)$. Transition $(s, a, s') \in \text{trans}(X)$ is written as $s \xrightarrow{a}_X s'$; we also write $s \rightarrow_X s'$ if a is internal. We define a relation $\rightarrow_X$ is the reflexive transitive closure of $\rightarrow_X$. For any $a \in \text{sig}(X)$ and $s, s' \in \text{states}(X)$, we write $s \xrightarrow{a} s'$ for $s \rightarrow_X s_1 \xrightarrow{a}_X s_2 \rightarrow_X s'$ with some $s_1, s_2 \in \text{states}(X)$ if a is external, or for $s \rightarrow_X s'$ if a is internal. For any $s_0 \in \text{start}(X)$ and transition sequence $\alpha \equiv s_0 \xrightarrow{a_1}_X s_1 \xrightarrow{a_2}_X \cdots \xrightarrow{a_n}_X s_n$, the trace

^{*} Corresponding author.

E-mail address: kawabe@theory.br1.ntt.co.jp (Y. Kawabe).

of α is the sub-sequence of $a_1a_2 \cdots a_n$ consisting of all the external actions. In addition, we write $traces(X)$ for the entire set of X 's traces.

3. Formalizing anonymity

This paper's approach to anonymity is based on the so-called "principle of confusion". That is, a system is anonymous if one user can cause a certain observable trace, then it is possible for the other users to cause the same trace (modulo special actions with regard to a user's identity).

3.1. Trace anonymity

Definition 1. Let X be an I/O-automaton and A be a family with the following conditions: (i) $\bigcup_{A' \in A} A' \subset ext(X)$; (ii) A' and A'' are disjoint for any distinct $A', A'' \in A$. We call A a *family of X 's actor actions*, and an element of $\bigcup_{A' \in A} A'$ is called an *actor action* (on A).

Actor actions are introduced by a protocol designer to discuss the anonymity of the protocol, while non-actor actions are employed to specify the body of the protocol. The occurrences of different actor actions should be indistinguishable to an adversary. This is formalized as follows.

Definition 2. Let X be an I/O-automaton and A be a family of X 's actor actions. We define I/O-automaton $anonym_A(X)$ as follows:

$$\begin{aligned} states(anonym_A(X)) &= states(X), \\ start(anonym_A(X)) &= start(X), \\ ext(anonym_A(X)) &= ext(X), \\ int(anonym_A(X)) &= int(X) \quad \text{and} \\ trans(anonym_A(X)) &= \left\{ (s_1, a, s_2) \mid (s_1, a, s_2) \in trans(X) \wedge a \notin \bigcup_{A' \in A} A' \right\} \\ &\quad \cup \left\{ (s_1, a, s_2) \mid (s_1, a', s_2) \in trans(X) \wedge A' \in A \wedge a' \in A' \wedge a \in A' \right\}. \end{aligned}$$

We say X is trace anonymous on A if

$$traces(anonym_A(X)) = traces(X)$$

holds.

Intuitively, $anonym_A(X)$ is anonymous in the sense that if $s \xrightarrow{\text{someone}} anonym_A(X) s'$ holds for some *someone* $\in$

```

automaton Jukebox
signature
  output startJB(qt:NonZeroNat, id:AorB),
    playMusic, playJazz, playRock

states
  pc:      PC      := start,
  quarter: NonZeroNat := 1

transitions
  output startJB(qt,id)
    pre pc = start
    eff quarter := qt;
      if id = Alice then pc := jazz
      else pc := rock fi

  output playMusic
    pre (pc = jazz /\ pc = rock)
      /\ quarter ~= 1
    eff quarter := quarter-1

  output playJazz      output playRock
    pre pc = jazz      pre pc = rock
      /\ quarter = 1   /\ quarter = 1
    eff pc := stop     eff pc := stop

```

Fig. 1. Specification of *Jukebox*.

A' with $A' \in A$ then $s \xrightarrow{\text{everyone}} anonym_A(X) s'$ holds for any $everyone \in A'$. If we have $traces(anonym_A(X)) = traces(X)$ then $anonym_A(X)$'s anonymity leads to X 's anonymity.

To explain trace anonymity, we consider a simple example. There is an electric jukebox in a building. If someone inserts n -quarters, the jukebox distributes the digital data of n -songs wirelessly. To listen to the music, people in the building use a device such as a PDA, which can receive and play the music data in real time. There are two people, Alice and Bob, and one of them is going to insert coins in the jukebox anonymously; namely, he/she does not want anyone to know who inserted the coins. He/she may select songs randomly, but for the last one, he/she always chooses a title from a favorite genre. Alice and Bob love jazz and rock music, respectively. Fig. 1 describes the above as I/O-automaton *Jukebox* in the IOA specification language [6]; its behavior is also depicted in Fig. 2. Fig. 1 has three portions: (i) *signature* declares actions; (ii) *states* declares variables and their initial value; (iii) *transitions* defines the body of actions, where each action consists of a precondition and an effect. In this example, we assume that the occurrence of $startJB(n, Alice)$ and that of $startJB(n, Bob)$ are indistinguishable to an adversary, and we employ $A = \{ \{ startJB(n, Alice), startJB(n, Bob) \} \mid n \in \{1,$

Download English Version:

<https://daneshyari.com/en/article/428435>

Download Persian Version:

<https://daneshyari.com/article/428435>

[Daneshyari.com](https://daneshyari.com)