Toward Formal Analysis of Timed Anonymous Systems

Yoshinobu Kawabe[1] and Nobuhiro Ito[1]
[1]Department of Information Science, Aichi Institute of Technology
Yachigusa 1247, Yakusa-cho, Toyota, Aichi 470-0392, Japan
E-mail : [1]{ kawabe, n-ito }@kwb.aitech.ac.jp

Abstract: This paper describes a basic idea to verify the anonymity of timed systems. Even though communication patterns are indistinguishable, the sender of a message can be identified by detecting the timing of message emission. In this paper we describe a timed system with an I/O-automaton-based formal specification language. By introducing a timer variable, we need to deal with an infinite-state system. With a simulation-based proof method for anonymity, we handle the infinite-state system directly.

Keywords—**Timed systems, Anonymity, Verification, Formal Method, I/O-automaton**

1. Introduction

Recently various real-time systems are used on the Internet. To establish the reliability of timed systems, there have been many studies based on formal methods that modeled and verified the correctness of timed systems [1][2]. In this paper we discuss anonymity, which is an important property with regard to privacy, of timed systems. We say a security protocol is anonymous if an adversary who can observe all the occurrences of events from the protocol cannot determine who is the "actor" of the events. There are many studies to describe and verify the anonymity of security protocols formally; for example, in [3] a proof technique that incorporates theorem-proving is introduced.

To establish anonymity, we should deal with patterns of communication such as the number of messages or the existence/nonexistence of a message. However, even though communication patterns are designed to be indistinguishable, the sender's identity may be disclosed by detecting a timing of message emission. Also, the sender's identity may be disclosed by detecting the occurrence of a timeout. That is, the detection of timing information may lead to the disclosure of who is an actor.

In this paper we describe a timed system with an I/O-automaton-based formal specification language [4]. This enables us to employ a proof method developed for the anonymity of untimed systems. By introducing a timer variable, we must deal with an infinite-state system. However, I/O-automaton theory [5][6] does not assume finiteness of the number of states or trace length, and it provides a proof technique called a simulation-based method that can handle infinite-state systems directly. In this paper we discuss how to apply a simulation-based method for proving the anonymity of timed systems.

This paper is organized as follows. We first present a simple motivating example in Section 2. Then, a timed system is described in IOA language in Section 3. After showing a basic idea for proving timed anonymity in Section 4, we have a discussion in Section 5.

2. Motivating Example: Sending Money

To explain the notion of timed anonymity, we introduce the following example.

Example 1: There are two people, Alice and Bob. Alice has \$50, while Bob has \$10,000. Charlie has requested only one of them to give him \$10. We do not know which person makes a payment, but one of them actually sends \$10.

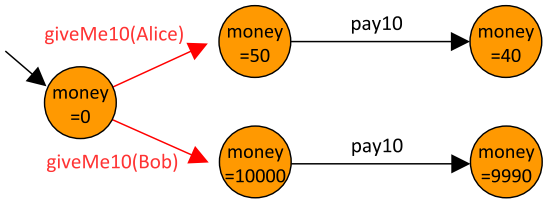

Figure 1. Automaton GMT

I/O-automaton GMT in Fig. 1 describes the above situation. Action giveMe10(*mem*), where *mem* is Alice or Bob, is a special action to represent the actor, and pay10 is an action for a payment. Automaton GMT has the trace set

$$traces(\mathtt{GMT}) = \left\{ \begin{array}{l} \mathtt{giveMe10(Alice).pay10}, \\ \mathtt{giveMe10(Bob).pay10} \end{array} \right\}.$$

In this case, an adversary who observed the occurrence of action pay10 cannot determine the preceding action. That is, both of giveMe10(Alice) and giveMe10(Bob) are possible, so the adversary never knows who made a payment. In [3], a system like GMT is called *trace anonymous*.

In Example 1, Alice possibly pays \$10 even though she has only \$50. In the following, we would like to consider a modified example.

Example 2: Bob has much money (\$10,000), so he can send \$10 immediately. But Alice has only \$50. When asked by Charlie, she thinks for a moment before sending \$10.

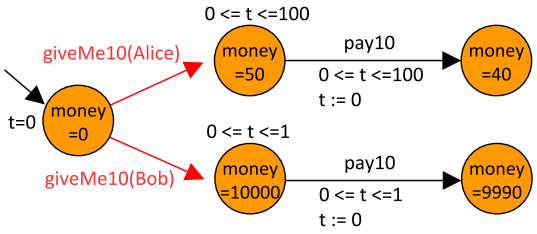

Figure 2. Timed Automaton GMTt

This is described with a timed automaton in Fig. 2. In this modeling, Alice who has only $50 may take some time up to 100 seconds before sending $10. On the other hand, Bob can make a decision within one second. From this observation, if the payment of $10 occurs after one second, then we can see the identity of the sender is Alice. This means that even though communication patterns are indistinguishable, the sender can be identified by detecting the timing of message emission.

3. Describing Timed Systems in IOA

This section describes a timed system with an (infinite-state) untimed version of I/O-automaton. With examples in the previous section, we explain a basic idea of timed anonymity.

IOA [4] is a formal specification language based on I/O-automaton theory. In IOA, a state is formalized as a tuple of values. Automaton GMT in Fig. 1 is written as follows.

```
automaton GMT
  signaure
    output giveMe10(mem: AorB)  % AorB
    output pay10                %  = { Alice, Bob }
  states
     money: Nat  := 0
  transitions
    output giveMe10(mem)        % This is an actor
      pre money = 0             % action.
      eff if (mem = Alice) then
            money := 50;
          else
            money := 10000;
          fi
    output pay10
      pre (money = 50 \/ money = 10000)
      eff money := money - 10
```

Here, two actions `giveMe10(mem)` and `pay10` are defined in a precondition-effect style; action `giveMe10(mem)` represents an actor of a computation . If a security protocol has an anonymous simulation [3], which is a binary relation over states, then the security protocol is trace anonymous. Actually, if we define a candidate binary relation $as_{\mathtt{GMT}}$ as:

$$as_{\mathtt{GMT}}(s, s') \iff s.\mathbf{money} = s'.\mathbf{money} \vee |s.\mathbf{money} - s'.\mathbf{money}| = 9950$$

then the binary relation satisfies the conditions to be an anonymous simulation of automaton GMT, where $\alpha.\beta$ represents the value of variable β at state α. Moreover, in this paper we introduce special variables:

- `timer` : a timer variable for elapsing time, and
- `timerFlg` : a flag variable for activating/deactivating the timer.

We can define the following automaton GMT2:

```
automaton GMT2
  signaure
    output giveMe10(mem: AorB)
    output pay10
    internal timerDeactivate
   states
     money: Nat  := 0,
     timer: Real  := 0.0,
     timerFlg: Bool  := true
   transitions
     output giveMe10(mem)
       pre money = 0 /\ ~timerFlg
       eff if (mem = Alice) then
             money := 50;
           else
             money := 10000;
           fi
           timerFlg := true
     output pay10
       pre (money = 50 \/ money = 10000)
           /\ ~timerFlg
       eff money := money - 10;
           timerFlg := true
     internal timerDeactivate   % This action is
       pre timerFlg             % internal and
       eff timerFlg := false    % does not appear
                                % in traces.
```

where we can easily see $traces(\mathtt{GMT2}) = traces(\mathtt{GMT})$. The value of `timerFlg` should be `false` if either `giveMe10(mem)` or `pay10` is enabled, and `timerFlg` becomes `true` if the action is actually fired. Also, action `timerDeactivate`, which is called a time action, is enabled only if `timerFlg` is `true` and it changes the value of `timerFlg` to `false`. This means that a normal action and a time action occur alternately. Note that action `timerDeactivate` does not change the value of `timer` and the time action does not appear in traces since it is internal.

By modifying GMT2, we can develop Fig. 2's automaton GMTt. Specifically, we remove `timerDeactivate` from GMT2 and we add the following three actions.

```
output giveMe10Time
  pre timerFlg /\ money = 0
  eff timerFlg := false

output pay10Time(t)
  pre    timerFlg /\ t = timer
      /\ (money = 50 \/ money = 10000)
      /\ ((money = 50) => (timer <= 100.0))
      /\ ((money = 10000) => (timer <= 1.0))
  eff timer := 0.0;
      timerFlg := false

output elapse(delta)
  pre    timerFlg /\ delta > 0
      /\ (money = 50 \/ money = 10000)
      /\ ((money = 50)
           => (   (timer <= 100.0)
               /\ (timer + delta <= 100.0)))
      /\ ((money = 10000)
           => (   (timer <= 1.0)
               /\ (timer + delta <= 1.0)))
  eff timer := timer + delta
```

Below we classify GMTt's actions as follows:

- Normal actions (`giveMe10(mem)` and `pay10`): appear in the original automaton GMT; and
- Time actions (`giveMe10Time`, `pay10Time(t)` and `elapse(delta)`) : are employed for expressing timing constraints and for elapsing time.

In GMTt, action `giveMe10Time` and its corresponding normal action `giveMe10(mem)` have a common condition "`money = 0`" in their precondition part. Also, `pay10Time(t)` and `pay10` have condition "`(money = 50 \/ money = 10000)`" in common. Moreover, actions `giveMe10Time` and `pay10Time(t)` do not rewrite variable `money`. Hence, after firing `giveMe10Time` or `pay10Time(t)`, its corresponding normal action is enabled. From this observation, we can

see that a one-step transition by action pay10 in Fig. 2 is formalized with a two-step transition sequence with pay10Time(t) and pay10 in IOA language.

Automaton GMTt has another time action, elapse(delta), and the output action is for elapsing time. The precondition of elapse(delta) defines the timing constraint at time timer and at time time + delta.

4. Analyzing Anonymity for Timed Systems

This section analyzes the anonymity of timed systems.

4.1 Counterexample for GMTt's anonymity

Automaton GMTt does not have a trace

```
giveMe10Time.giveMe10(Bob).
    elapse(30).pay10Time(30).pay10
```

that represents "Bob is asked and he pays $10 after 30 seconds"; note that Bob must make a payment in one second. However, GMTt's corresponding anonymous system $anonym_{\{\{\texttt{Alice,Bob}\}\}}(\texttt{GMTt})$ has the above trace; that is, we cannot say $traces(\texttt{GMTt}) = traces(anonym_{\{\{\texttt{Alice,Bob}\}\}}(\texttt{GMTt}))$. This means that $anonym_{\{\{\texttt{Alice,Bob}\}\}}(\texttt{GMTt})$'s anonymity does not lead to GMTt's anonymity. Therefore, GMTt is not anonymous.

4.2 Anonymizing GMTt

In this section we modify GMTt. Specifically, we define:

```
output pay10Time(t)
  pre    timerFlg /\ t = timer
      /\ (money = 50 \/ money = 10000)
      /\ ((money = 50) => (timer <= 1.0))
      /\ ((money = 10000) => (timer <= 1.0))
  eff timer := 0.0;
      timerFlg := false

output elapse(delta)
  pre    timerFlg /\ delta > 0.0
      /\ (money = 50 \/ money = 10000)
      /\ ((money = 50)
           => (   (timer <= 100.0)
               /\ (timer + delta <= 1.0)))
      /\ ((money = 10000)
           => (   (timer <= 1.0)
               /\ (timer + delta <= 1.0)))
  eff timer := timer + delta
```

for pay10Time(t) and elapse(delta). That is, we replace conditions "timer <= 100.0" and "timer + delta <= 100.0" in pay10Time(t) and elapse(delta) with "timer <= 1.0" and "timer + delta <= 1.0", respectively. We call the resulting automaton GMTt2. This is to assume Alice responds in one second.

The modified automaton has an anonymous simulation:

$$as_{\texttt{GMTt2}}(s,s') \iff as_{\texttt{GMT}}(s,s') \wedge s.\texttt{timer} = s'.\texttt{timer} \wedge (s.\texttt{timerFlg} \iff s.\texttt{timerFlg}).$$

This formula contains GMT's anonymous simulation relation $as_{\texttt{GMT}}$. With this relation, we can prove the anonymity of GMTt2 with the following steps:

1. Find an anonymous simulation for GMT;
2. Then, extend the anonymity result for GMTt2.

In the remainder of this section, we describe why the above proof is possible.

4.2.1 GMTt2's initial state's condition

Let $(s,t,p) \in start(\texttt{GMTt2})$ be an initial state of GMTt2, where s is a tuple that represents a state of automaton GMT, t is a value of variable timer, and p is a value of variable timerFlg. From the definition of GMTt2, we have $t = 0.0$ and $u = \texttt{true}$. Clearly, $as_{\texttt{GMT}}(s,s)$ implies $as_{\texttt{GMTt2}}((s,0.0,\texttt{true}),(s,0.0,\texttt{true}))$.

4.2.2 Step's correspondence for normal actions

A normal action of GMTt2 can be enabled only if the value of variable timerFlg is false. If the action is fired, then variable timerFlg is changed to be true, but timer is not changed. Hence, we can see that for any normal action a we have:

- $(s_1,t,p) \xrightarrow{a}_{\texttt{GMTt2}} (s_1',t,p')$ and
- $as_{\texttt{GMTt2}}((s_1,t,p),(s_2,u,q))$

implies

- We have $t = u$ from the definition of $as_{\texttt{GMTt2}}$;
- We have $p = q = \texttt{false}$ and $p' = \texttt{true}$ since a is a normal action; and
- We have $as_{\texttt{GMT}}(s_1,s_2)$ and $s_1 \xrightarrow{a}_{\texttt{GMT}} s_1'$.

Thus, there exists a state s_2' of GMT such that:

- We have $s_2 \stackrel{a'}{\Longrightarrow}_{\texttt{GMT}} s_2'$ and $as_{\texttt{GMT}}(s_1',s_2')$;
- $a \in \{\texttt{giveMe10(Alice)},\texttt{giveMe10(Bob)}\}$ implies $a' \in \{\texttt{giveMe10(Alice)},\texttt{giveMe10(Bob)}\}$; and
- $a = \texttt{pay10}$ implies $a' = a = \texttt{pay10}$.

Therefore, for the state $(s_2',t,\texttt{true})$, we have:

- $(s_2,u,q) \equiv (s_2,t,\texttt{false}) \stackrel{a'}{\Longrightarrow}_{\texttt{GMTt2}} (s_2',t,\texttt{true})$, and
- $as_{\texttt{GMTt2}}((s_1',t,p'),(s_2',t,\texttt{true}))$.

Consequently, if binary relation $as_{\texttt{GMT}}$ is an anonymous simulation, then binary relation $as_{\texttt{GMTt2}}$ satisfies a step correspondence condition for any normal action.

4.2.3 Step's correspondence for time actions

If a time action is enabled at a state, then the value of timerFlg is true. Also, variables timer and timerFlg can be changed by the time action. Hence, for any time action b, we have:

- $(s_1,t,p) \xrightarrow{b}_{\texttt{GMTt2}} (s_1',t',p')$, and
- $as_{\texttt{GMTt2}}((s_1,t,p),(s_2,u,q))$

implies

- $s_1' = s_1$, $t = u$, and $p = q = \texttt{true}$ holds;
- If b is elapse(delta) then $p' = \texttt{true}$; otherwise, $p' = \texttt{false}$; and
- $as_{\texttt{GMT}}(s_1,s_2)$ holds.

If we can prove

$$(s_2,u,q) \equiv (s_2,t,\texttt{true}) \stackrel{b}{\Longrightarrow}_{\texttt{GMTt2}} (s_2,t',p')$$

for state (s_2,t',p'), then $as_{\texttt{GMTt2}}((s_1,t',p'),(s_2,t',p'))$ holds. Hence, $as_{\texttt{GMTt2}}$ satisfies the conditions to be an anonymous simulation of GMTt2 for action b.

4.3 Further analysis for GMTt2

We consider the transition

$$(s_1, t, p) \equiv (s_1, t, \texttt{true}) \xrightarrow{b}_{\texttt{GMTt2}} (s_1, t', p') \equiv (s_1', t', p')$$

shown in the previous section and a transition $(s_2, t, \texttt{true}) \xrightarrow{b}_{\texttt{GMTt2}} (s_2, u', q')$ by time action b. From the definition of each time action, we have $u' = t'$ and $q' = p'$. Moreover, the condition sequence

$$(s_2, u, q) \equiv (s_2, t, \texttt{true}) \stackrel{b}{\Longrightarrow}_{\texttt{GMTt2}} (s_2, t', p')$$

is actually a one-step transition

$$(s_2, u, q) \equiv (s_2, t, \texttt{true}) \xrightarrow{b}_{\texttt{GMTt2}} (s_2, t', p')$$

since GMTt2 does not have any internal actions. Hence, for GMTt2, we can prove the anonymity by proving:

> For any GMTt2's time action b and any states s_1, s_2 with $as_{\texttt{GMT}}(s_1, s_2)$, if action b is enabled at state (s_1, t, p) then b is also enabled at (s_2, u, q).

Specifically, it suffices to show the following three formulae with a theorem proving tool [7], where `enabled(s, a)` is true if action a is enabled at state s:

```
(as(s1, s2) /\ enabled(s1, giveMe10Time))
 => enabled(s2, giveMe10Time),
(as(s1, s2) /\ enabled(s1, pay10Time(t)))
 => enabled(s2, pay10Time(t))
```

and

```
(as(s1, s2) /\ enabled(s1, elapse(delta)))
 => enabled(s2, elapse(delta)).
```

5. Discussion

In this study, we described a timed system as an infinite-state system with a conventional I/O-automaton, and we applied the proof method for anonymity [3] directly. As another approach, it seems possible to redefine the anonymity proof technique of [3][10] in timed automaton [8] or in timed I/O-automaton [9]. In this section we compare the approaches.

Timed automaton models are designed for dealing with timing features of computation; so, several constraints are introduced to verify timing properties properly. For example, an execution sequence where only time actions occur infinitely often and normal actions do not occur is regarded as unfair, and unfair execution sequences are usually prohibited. However, for anonymity verification we may not need such a condition; even though there is an unfair execution sequence by actor Alice in a security protocol, we can discuss the anonymity if the security protocol has another corresponding execution sequence by actor Bob.

The untimed I/O-automaton model does not support such conditions, but it has various verification tools and proof methods, and we can use them to prove anonymity. This as an advantage of using untimed I/O-automaton theory. However, in our approach we should introduce a parameter for real-valued times; that is, we must handle infinite-state systems. We can overcome this problem since I/O-automaton theory [5][6] does not assume finiteness of the number of states or trace length, and simulation-based proof techniques are applicable to prove the trace inclusion of infinite-state systems.

We compared the both approaches, and in this study we employed a formal specification language based on conventional I/O-automaton theory. The main reason is that various verification tools are available.

6. Conclusion

This paper discussed a method to verify the anonymity of timed systems. By describing a timed system with an I/O-automaton-based formal specification language, a proof technique for anonymity of untimed systems can be applied to a timed system.

This paper has shown a basic idea to prove the anonymity of timed systems with a small example. It is a future work to deal with a larger example such as Mixnet [11].

Acknowledgment

This study is supported by the Grant-in-Aid for Scientific Research (C), No.26330166, of the Ministry of Education, Culture, Sports, Science and Technology, Japan.

References

[1] K. van Hee and N. Sidorova, "The Right Timing: Reflections on the Modeling and Analysis of Time", *PETRI NETS 2013*, LNCS 7927, pp.1-20, Springer, 2013.

[2] M. Wehrle and S. Kupferschmid, "Mcta: Heuristics and Search for Timed Systems", *FORMAT 2012*, LNCS 7595, pp.252-266, Springer, 2012.

[3] Y. Kawabe, K. Mano, H. Sakurada and Y. Tsukada, "Theorem-proving anonymity of infinite-state systems". *Inf. Proc. Lett.*, vol. 101, no. 1, pp. 46–51, 2007.

[4] A. Bogdanov, "Formal verification of simulations between I/O-automata", Master's thesis, MIT, 2000.

[5] N. A. Lynch and F. Vaandrager, "Forward and backward simulations — part I: Untimed systems". *Inform. and Comput.*, Vol. 121, No. 2, pp. 214-233, 1995.

[6] N. A. Lynch, *Distributed algorithms*, Morgan Kaufmann Publishers, 1996.

[7] J. F. Soegaard-Andersen, S. J. Garland, J. V. Guttag, N. A. Lynch, and A. Pogosyants. "Computer-assisted simulation proofs". In *CAV '93*, LNCS 697, pp. 305-319. Springer-Verlag, 1993.

[8] R. Alur and D. Dill, "A theory of timed automata". *TCS*, Vol. 126, pp. 183-235, 1994.

[9] D. Kaynar et.al, "The Theory of Timed I/O Automata". *Synthesis Lectures on Computer Science*, Morgan Claypool Publishers, 2010.

[10] I. Hasuo, Y. Kawabe and H. Sakurada, "Probabilistic anonymity via coalgebraic simulations". *TCS*, Vol. 411, No. 22-24, pp. 2239-2259, 2010.

[11] D. L. Chaum, "Untraceable electronic mail, return addresses, and digital pseudonyms". *CACM*, Vol. 24, No. 2, pp. 84-90, 1981.