SCIS 2006 The 2006 Symposium on Cryptography and Information Security
Hiroshima, Japan, Jan. 17-20, 2006
The Institute of Electronics, Information and Communication Engineers

Role Interchangeability and Verification of Electronic Voting

Ken Mano * Yoshinobu Kawabe * Hideki Sakurada * Yasuyuki Tsukada *

Abstract— We propose a new information-hiding property called *role interchangeability*. We specify the property in multiagent systems, and describe its relationship with known anonymity and privacy properties. Then a method is presented to prove role interchangeability of a protocol described as an automaton. As a case study, we apply our method to the formal verification of the FOO electronic voting protocol.

Keywords: security protocol, formal verification, electronic voting, anonymity, privacy.

1 Introduction

We propose a new information-hiding property called role interchangeability and present its theoretical foundation. Role interchangeability means that any two agents could interchange their *roles*, that is, the actions they performed.

We have two objectives, the first of which is to fill the *gap* between logic and computation in the formal verification of anonymity and privacy. Epistemic logics and formal computation models have been two classical approaches to such verification. However, the relationship between them hasn't been fully studied. Researches have recently been undertaken on "bridging" the gap [4, 5]. However, these results cannot be directly applied to the verification of electronic voting. To verify practical protocols, it is needed to make this "bridge" wider, and we achieve this by formalizing role interchangeability in multiagent systems and by providing a simulation proof method for the property.

Our second objective is to reveal the *duality* between anonymity and privacy. There have been some researches on the formal specification and verification of privacy [1, 5]. However, as far as we know, every known formal definition of privacy depends on some specific problem and so is not general enough. In fact, it is difficult to provide such a formal and general definition since casual meaning of privacy includes various issues. In this paper, we define privacy as some kind of dual of anonymity. Moreover, we show that exploiting this duality is useful for deriving anonymity and privacy from role interchangeability.

This paper is organized as follows. In Section 2, we introduce role interchangeability and show its properties. We specify role interchangeability in multiagent systems [4] and show that it derives various known anonymity properties under appropriate conditions. We then present a duality property that is useful for deriving privacy. In Section 3 we show a way to prove role interchangeability. First we introduce the ω-compatibility of an interpreted system with a set of traces. Based on this, role interchangeability is characterized by the existence of role-interchange functions on traces. We then present a simulation proof method to prove the existence of role-interchange functions for a given automaton. In order to examine the applicability of the verification method, in Section 4 we prove the anonymity and privacy of a practical electronic voting protocol. The protocol is described as an automaton and shown to have role-interchange functions.

* NTT Communication Science Laboratories, NTT Corporation. 3-1 Morinosato Wakamiya Atsugi-shi Kanagawa, 243-0198 Japan.

2 Specification by Multiagent Systems

We specify role interchangeability and present its properties in multiagent systems, which is a modal logic system with operator K_i to represent the knowledge of agent i. Notions and terminologies are borrowed from [4].

2.1 Role Interchangeability

Let I be the set of all *agents*, and A be the set of all *agent actions*. We assume $i, i', i_1, i_2, \ldots$ range over agents while $a, a', a_1, a_2, \ldots$ range over agent actions. Following [4], we use the formula $\theta(i, a)$ to represent "agent i has performed an agent action a, or will perform a in the future"[1].

Definition 2.1 Set A of agent actions is *role interchangeable* with respect to agent j in the interpreted system $\mathcal{I}$ if the following holds for any $i, i' \in I$ and $a, a' \in A$:

$$\mathcal{I} \models (\theta(i,a) \wedge \theta(i',a')) \Rightarrow P_j[\theta(i',a) \wedge \theta(i,a')],$$

where $P_j\phi$ is an abbreviation of $\neg K_j \neg \phi$, meaning that "agent j thinks that ϕ is possible".

Role interchangeability means that, as far as agent j is concerned, any two agents could interchange their *roles*, that is, the actions they performed.

We compare this property with a series of anonymity properties appearing in [4].

[1] An agent action in this paper is simply called an 'action' in [4]. We changed the terminology since we must distinguish the actions of agents from those of automata appearing in the next section.

Definition 2.2 Agent action a, performed by agent i, is *minimally anonymous* with respect to agent j in the interpreted system $\mathcal{I}$ if $\mathcal{I} \models \neg K_j[\theta(i,a)]$.

Agent action a, performed by agent i, is *totally anonymous* with respect to j in the interpreted system $\mathcal{I}$ if $\mathcal{I} \models \theta(i,a) \Rightarrow \bigwedge_{i' \neq j} P_j[\theta(i',a)]$.

Intuitively, minimal anonymity means agent j never knows that action a is performed by agent i, while total anonymity means, from j's viewpoint, a could have been performed by anybody in the system except j itself.

Basically, role interchangeability and the above anonymities are incomparable. However, minimal anonymity is derived from role interchangeability by assuming certain conditions:

Theorem 2.3 *Assume the following conditions hold for $i \in I$ and $a \in A$.*

1. $\mathcal{I} \models \theta(i,a) \Rightarrow \bigvee_{i' \in I} \bigvee_{a' \in A} [i \neq i' \wedge \theta(i',a')]$
2. $\mathcal{I} \models \bigwedge_{i \neq i'} \neg[\theta(i,a) \wedge \theta(i',a)]$

If A is role interchangeable in $\mathcal{I}$ with respect to j, then action a performed by agent i is minimally anonymous as well. ∎

Below is another version of anonymity that generalizes total anonymity so that an *anonymity set* $I_A \subseteq I$ can be used instead of I.

Definition 2.4 Agent action a, performed by agent i, is *anonymous up to* $I_A \subseteq I$ with respect to j if

$$\mathcal{I} \models \theta(i,a) \Rightarrow \bigwedge_{i' \in I_A} P_j[\theta(i',a)].$$

For an appropriate I_A, anonymity up to I_A follows from role interchangeability.

Theorem 2.5 *Let I_A be the set of agents that perform an agent action in all runs in $\mathcal{I}$, that is, the set*

$$\{i \in I \mid \forall (r,m) \in \mathcal{R}(\mathcal{I})\ \exists a \in A\ (\mathcal{I}, r, m) \models \theta(i,a)\}.$$

If A is role interchangeable in $\mathcal{I}$, then agent action $a \in A$ performed by $i \in I$ is anonymous up to I_A. ∎

Corollary 2.6 *Suppose that any agent in I performs an agent action in A. If A is role interchangeable in $\mathcal{I}$, then agent action a performed by i is totally anonymous.* ∎

2.2 Duality and Privacy

Here, we present some duality properties with respect to θ. Let us consider duality to replace I with A, A with I, and θ with θ' satisfying $\forall i \in I\ \forall a \in A\ \theta'(a,i) \Leftrightarrow \theta(i,a)$. The formula $\theta'(a,i)$ is read as "agent action a has been performed, or will be performed in the future by an agent i", so this is the duality of the passive voice.

Then, both role interchangeability and minimal anonymity are equivalent to its dual. Thus, for deriving minimal anonymity from role interchangeability, we can assume conditions 1' and 2' below instead of 1 and 2 in Theorem 2.3.

1'. $\mathcal{I} \models \theta(i,a) \Rightarrow \bigvee_{a' \in A} \bigvee_{i' \in I} [a \neq a' \wedge \theta(i',a')]$

2'. $\mathcal{I} \models \bigwedge_{a \neq a'} \neg[\theta(i,a) \wedge \theta(i,a')]$

Suppose $A = \{vote(1), \ldots, vote(cmax)\}$, where $cmax$ is a natural number. Assume that the intended interpretation of $\theta(i, vote(j))$ is that voter i voted for candidate j. Then, condition 2 means that no two voters vote for the same candidate, which is quite unnatural as regards voting. On the other hand, 2' is a much more adequate condition, namely that a voter does not vote for two candidates. Although this adequacy depends on the interpretation, the example shows that this duality is useful in terms of obtaining appropriate premises for the problem.

Let us next consider anonymity up to I_A. This time it is not equivalent to its dual, but the dual itself seems to be an interesting information-hiding property:

$$\mathcal{I} \models \theta(i,a) \Rightarrow \bigwedge_{a' \in A_I} P_j[\theta(i,a')],$$

where $A_I \subseteq A$. This property intuitively means that agent i could have performed any agent action in A_I, and can be thought of as a privacy of i. In other words, hiding *who* performs the agent action is anonymity while hiding *what* is performed by the agent is privacy. Thus we call it *privacy up to* A_I. Formally, agent i performing agent action a is private up to A_I with respect to j if the above condition holds. The following is the dual of Theorem 2.5.

Theorem 2.7 *Let A_I be the set of agent actions that is performed by an agent in all runs in $\mathcal{I}$, that is, the set $\{a \in A \mid \forall (r,m) \in \mathcal{R}(\mathcal{I})\ \exists i \in I\ (\mathcal{I}, r, m) \models \theta(i,a)\}$. Then, if A is role interchangeable in $\mathcal{I}$, then agent $i \in I$ performing $a \in A$ is private up to A_I.* ∎

3 Verification by Automata

3.1 ω-Compatibility and Role-Interchange Function

ω-compatibility is a variant of the compatibility introduced by Halpern and O'Neill to characterize anonymity up to I_A by strong anonymity [9] for CSP. Our notion modifies the original by introducing a *view* function. Moreover, the original notion implicitly assumes finiteness of trace length while our notion does not.

Let I be the set of all agents and A be the set of all agent actions as in the previous section. Σ is the set of all *trace actions* (or *actions*, simply). Let $\Sigma_{I,A} = \{i.a \mid i \in I, a \in A\}$ and suppose $\Sigma_{I,A} \subseteq \Sigma$. A finite or infinite sequence of actions in Σ is called a *trace* on Σ. An action $i.a \in \Sigma_{I,A}$ is an "imaginary" action to make a trace reflect the fact that $\theta(i,a)$ holds in the corresponding run. Thus, such actions are not intended to be observed directly.

Then we define a notion of *view function*. Let α be a fresh constant that represents an "invisible" action.

Definition 3.1 Let f_V be a function whose domain is Σ. If f_V satisfies the following property, it is called a *view function*:

$$\forall x \in \Sigma_{I,A}\ f_V(x) = \alpha.$$

The domain of a view function is homomorphically extended to traces and sets of traces.

We can use a view function to represent the ability of the observer. For instance, let $send(\sigma_k(data))$ be an action to send message $\sigma_k(data)$ encrypted with a secret key k that only the sender knows. Then, using a fresh constant *envelop*, the invisibility of the encrypted message can be modeled by defining the view function as $\forall x\ f_V(send(\sigma_k(x))) = send(envelop)$.

In the following, we suppose the existence of two special agents: the environment e and the observer o. A prefix of a trace t of length n is denoted by $t|n$. For convenience, for a trace t of length less than n we define $t|n = t$.

Definition 3.2 Let f_V be a view function. We say that a system $\mathcal{R}$ is *ω-compatible* with a set T of traces via a view function f_V if the following two conditions hold:

1. for every run $r \in \mathcal{R}$, there is a trace $t \in T$ such that for every time m, $t|m = r_e(m)$ and $f_V(t|m) = r_o(m)$, and
2. for every trace $t \in T$, there is a run $r \in \mathcal{R}$ such that for every time m, $r_e(m) = t|m$ and $r_o(m) = f_V(t|m)$.

Moreover, an interpreted system $\mathcal{I} = (\mathcal{R}, \pi)$ is ω-compatible with T if $\mathcal{R}$ is ω-compatible with T and if $(\mathcal{I}, r, m) \models \theta(i, a)$ whenever the action $i.a$ is in the trace $r_e(m')$ for some m'.

Then we introduce the role-interchange functions of a trace set. The existence of the functions characterizes the role interchangeability of the system ω-compatible with the set.

Definition 3.3 Let f_V be a view function and T be a set of traces on Σ. Let i, i' be any distinct agents, and a, a' be any distinct agent actions.

A function $f_{i,i'}^{a,a'} : T \to T$ is called a *role-interchange function* of T with respect to i, i' and a, a' via f_V if the following two conditions hold for any $t \in T$:

1. $f_V(t) = f_V(f_{i,i'}^{a,a'}(t))$.
2. If $i.a$ and $i'.a'$ appear in t, then $i'.a$ and $i.a'$ appears in $f_{i,i'}^{a,a'}(t)$.

Theorem 3.4 *Suppose an interpreted system $\mathcal{I} = (\mathcal{R}, \pi)$ is ω-compatible with a set T of traces via a view function f_V. Then the set A of agent actions is role interchangeable in $\mathcal{I}$ if and only if there is a family $\{f_{i,i'}^{a,a'} \mid$ distinct $i, i' \in I$, distinct $a, a' \in A\}$ of role interchange functions of T via f_V.* ∎

3.2 Simulation Proof Method

Given an automaton formalizing the behavior of a system, we present a sufficient condition for the trace set of an automaton to have role-interchange functions. This condition guarantees the existence of a family $\{f_{i,i'} | i, i' \in I\}$ of role-interchange functions that are independent of agent actions.

An *automaton* is a quadruple consists of sets of states, actions, initial states and transitions. Unlike a usual definition, we do not use a set of final states since it is not important for describing the system behavior. A transition is a triple (s, δ, s') of states s, s' and action δ, and is also written as $s \xrightarrow{\delta} s'$. Let $S = (States, Actions, Start, Trans)$ be an automaton.

An *execution* of S is a finite or infinite alternating sequence $s_0, \delta_1, s_1, \delta_2, s_2, \ldots$ of states and actions that satisfy $s_0 \in Start$ and $(s_i, \delta_{i+1}, s_{i+1}) \in Trans$ for every i . The set of all executions of S is denoted by $execs(S)$. The sequence obtained by removing all states from an execution e is called a *trace* of e, denoted by $trace(e)$. We write $traces(S)$ instead of $trace(execs(S))$.

In the second condition of a role-interchange function, agents i and i' are assumed to perform actions in $\Sigma_{I,A}$ in a given trace. The following is the notion that corresponds to the assumption for executions.

Definition 3.5 Let i and i' be arbitrary distinct agents. An execution is called *coexistable* for i and i' if there are agent actions a and a' such that $i.a$ and $i'.a'$ appear in the execution. A state in a coexistable execution for i and i' is called a coexistable state for i and i'.

Then we present a method for constructing role-interchange functions by the stepwise simulation of executions.

Definition 3.6 (Agent-exchange function on coexistable states) Let S be an automaton, I be the set of all agents, A be the set of all agent actions, and f_V be a view function. An *agent exchange function* $g_{i,i'}$ of S with respect to distinct agents $i, i' \in I$ via f_V is a function on states that satisfy the following conditions for any coexistable states s, s' for i and i':

- For any $a \in A$, $s \xrightarrow{i.a} s'$ implies $g_{i,i'}(s) \xrightarrow{i'.a} g_{i,i'}(s')$.
- For any $a \in A$, $s \xrightarrow{i'.a} s'$ implies $g_{i,i'}(s) \xrightarrow{i.a} g_{i,i'}(s')$.
- For any action δ not of the form $i.a$ or $i'.a$, $s \xrightarrow{\delta} s'$ implies there is an action δ' s.t. $g_{i,i'}(s) \xrightarrow{\delta'} g_{i,i'}(s')$ and $f_V(\delta) = f_V(\delta')$.

Theorem 3.7 *Let S be an automaton, I be the set of all agents, and f_V be a view function. Suppose there is an agent-exchange function of S with respect to i, i' via f_V for any distinct agents $i, i' \in I$. Then $traces(S)$ has a family $\{f_{i,i'} \mid i, i' \in I\}$ of role-interchange functions via f_V.* ∎

4 Case Study: Electronic Voting

4.1 FOO Protocol

In this section we explain an electronic voting protocol called FOO [3] and present some assumptions for verifying its anonymity and privacy. Formal anonymity verifications for FOO have been presented in [7] and [6]. We aim at providing a logical foundation for such simulation-based verifications. The model of FOO involves the following agents: voters $V_1, \ldots, V_{vmax}$, an

administrator Ad, and a collector Co. The administrator verifies that only eligible voters can cast votes, and the collector collects and publishes the votes.

We assume that readers are familiar with the properties of basic cryptographic primitives such as a *commitment* $\xi(v, r)$, a *digital signature scheme* σ_x for agent x, a *blind signature scheme* σ'_{Ad} with a *blinding factor* χ, formalized by the following equations:

$$\xi^{-1}(\xi(v, r), r) = v,$$
$$\sigma_{Ad}^{-1}(\sigma_{Ad}(m)) = m,\ \sigma_i^{-1}(\sigma_i(m)) = m,$$
$$\chi^{-1}(\sigma'_{Ad}(\chi(m, b)), b) = \sigma_{Ad}(m).$$

We also assume $\sigma'_{Ad}(x) \neq null$.

The FOO protocol consists of 3 phases:

[phase 1] Voter V_i to vote for v_i computes a ballot $x_i = \xi(v_i, r_i)$ using key r_i, and the message $e_i = \chi(x_i, b_i)$. Then V_i sends the administrator Ad the signed message $\sigma_i(e_i)$ together with his identifier i. Then Ad confirms the signature and verifies that it is the first vote by an eligible voter. If it passes the tests, Ad replies to v_i with the signed message $\sigma'_{Ad}(e_i)$. At the end of the phase, Ad publishes the list of all authenticated voters with their blinded ballots.

[phase 2] V_i sends the collector Co the message $\sigma_{Ad}(x_i)$ obtained by unblinding the reply from Ad via a *sender-anonymous communication channel* such as the mixnet [2]. Then Co confirms the signature of Ad. If it does not pass the test, Co claims this by showing the bad signature. At the end of the phase, Co publishes the pair $(l_i, \sigma_{Ad}(x_i))$ for every vote that passed the test, where l_i is a number unique to each vote.

[phase 3] V_i sends Co the message (l_i, r_i) again via the sender-anonymous channel. Co checks that r_i is a correct key for x_i, and counts and announces the voting result at the end of the phase.

We assume that a voter may abstain from voting, so we omit the number check of voters found in phase 3 of the original protocol. Furthermore, we assume that only V_i knows b_i, which affects the definition of the view function in the next section.

4.2 Automaton Description of FOO

We describe the behavior of FOO in the precondition-effect style. This style is based on that of *I/O-automata* in [8]. Actually the description in this section can be thought of as an I/O-automaton with only output actions.

Conceptually, the system consists of the agents introduced in the previous section together with a special agent called *phase manager*. Formally, the system is an automaton S whose actions are observed by the attacker from his viewpoint. A state of S consists of fields *phase*, *abuff*, *line*, *vbuff*, *kbuff* and *voter*(i) $(i = 1, \ldots, vmax)$, each of which is associated with an agent as shown in Figure 1. The automaton S performs actions *phase*2, *phase*3, *auth*, *ack*, etc., and

For simplicity, we assume the following:

1. *phase* can be referred from anyone at anytime. Also, each communication between a voter and the administrator/collector is completed immediately.
2. The attacker is an eavesdropper.

Moreover, all message checks and claim processing are omitted, and the administrator returns a reply to all requests without any eligibility check.

Constants, Functions, and Primitive Propositions. Constants Ph, Ad, Co, and natural numbers $1, \ldots, vmax$ are used to represent the phase manager, the administrator, the collector, and voters $1, \ldots, vmax$, respectively. Candidates are also represented by natural numbers $1, \ldots, cmax$, and they are distinguished from voters by the context. For multisets to represent buffers, we use the multiset constructor -:, the empty multiset $\emptyset$, the multiset membership $\in$, and the function $delete(x, y)$ that deletes an occurrence of x in y if any exist. It is easy to formalize the properties of the data structure used here by using first-order formulae with equations.

States. A state consists of fields and subfields. Flags *authed*, *casted*, *keyed* and $finish$ range over boolean constants initially $false$, and buffers *abuff*, *vbuff* and *kbuff* are multisets initially $\emptyset$. Fields and subfields are designated using _._. For instance, subfield X of field Y in state s is denoted by $s.Y.X$. The prefix $s.$ is often omitted when the state s is clear from the context.

Actions. Each action other than *vote* corresponds to a communication between agents, and has the form $action\_name(sender, receiver, data_1, data_2, \ldots)$. We suppose that *null* receiver means the action corresponds to a publication. Phase synchronizations are formulated as a publication from the phase manager.

$vote(i, j)$ should be written as $i.vote(j)$ in the notation of Section 3, where $I = \{1, \ldots, vmax\}$ and $A = \{vote(null), vote(1), \ldots, vote(cmax)\}$. The action reflects i's voting behavior; performing $vote(i, j)$ for $j = 1, \ldots, cmax$ implies that i voted for j, while $vote(i, null)$ means that i received the voting right from Ad but did not actually cast a vote.

Transition rules. The transition rules are described in precondition-effect style as shown in Figure 2. States are implicit in the description. For instance, the $phase2$ rule is read as "for any state s, $phase2(Ph, null)$ action is possible when $s.phase = 1$, and if the action is performed, in the next state s' the value of $s'.phase$ will be updated to 2".

Note that each phase can be shut before all voters finish their actions in the phase. If phase 1 is shut before voter i performs *ack* action, then he will not continue the following actions. If phase 2 is shut before i's *pub* action, i can then perform only $vote(i, null)$.

4.3 Role-Interchange Function for FOO

The definition of view function f_V reflects the assumptions that the observer does not know the blinding factors $voter(i).b$ and that each voter sends a message to the collector via the sender-anonymous channel.

Definition 4.1 f_V is a view function defined such that

$$f_V(auth(i, Ad, \sigma_i(x))) = auth(i, Ad, \sigma_i(env(i)))$$
$$f_V(ack(i, Ad, \sigma'_{Ad}(x))) = ack(i, Ad, \sigma'_{Ad}(env(i)))$$
$$f_V(cast(x, Co, svote)) = cast(null, Co, svote)$$
$$f_V(key(x, Co, l, r)) = key(null, Co, l, r)$$
$$f_V(vote(x, y)) = \alpha$$

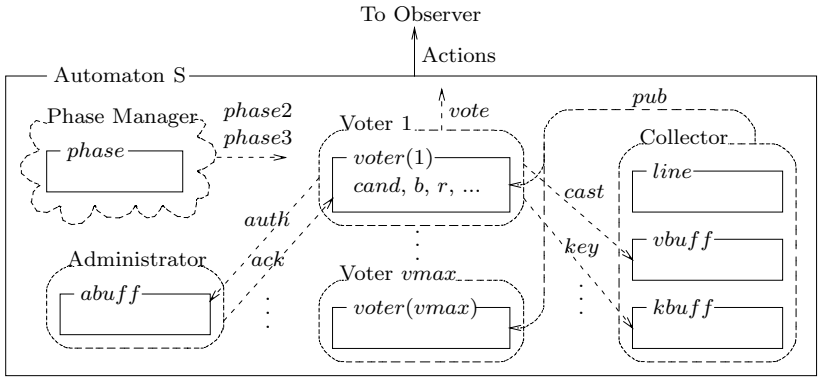

Figure 1: Conceptual model of electronic voting system

$phase2(Ph, null)$
Precondition:
$phase = 1$
Effect:
$phase := 2$

$phase3(Ph, null)$
Precondition:
$phase = 2$
Effect:
$phase := 3$

$auth(i, Ad, req)$
Precondition:
$phase = 1 \wedge voter(i).authed = false \wedge$
$req = \sigma_i(voter(i).cand)$
Effect:
$voter(i).authed := true$
$abuff := pair(i, req)$ -: $abuff$

For any Γ,
$ack(Ad, i, sreq)$
Precondition:
$phase = 1 \wedge pair(i, \Gamma) \in abuf \wedge$
$sreq = \sigma'_{Ad}(\sigma_i^{-1}(\Gamma))$
Effect:
$abuff := delete(pair(i, \Gamma), abuff)$
if
$voter(i).authed = true$
then
$voter(i).sreq := sreq$
fi

$cast(i, Co, svote)$
Precondition:
$phase = 2 \wedge voter(i).sreq \neq null \wedge$
$voter(i).casted = false \wedge$
$svote = \chi^{-1}(voter(i).sreq, voter(i).b)$
Effect:
$voter(i).casted := true$
$vbuff := svote$ -: $vbuff$

$pub(Co, null, l, svote)$
Precondition:
$phase = 2 \wedge svote \in vbuff \wedge l = line$
Effect:
$vbuff := delete(svote, vbuff)$
$line := line + 1$
For any $\theta = 1, \ldots, vmax$,
if
$voter(\theta).casted = true \wedge$
$svote = \chi^{-1}(voter(\theta).sreq, voter(\theta).b)$
then
$voter(\theta).line := l$
fi

$key(i, Co, l, r)$
Precondition:
$phase = 3 \wedge voter(i).line \neq null \wedge$
$voter(i).keyed = false \wedge$
$l = voter(i).line \wedge r = voter(i).r$
Effect:
$voter(i).keyed := true$
$kbuff := pair(l, r)$ -: $kbuff$

$vote(i, null)$
Precondition:
$phase = 3 \wedge voter(i).sreq \neq null \wedge$
$voter(i).line = null \wedge voter(i).finish = false$
Effect:
$voter(i).finish := true$

$vote(i, j)$
Precondition:
$phase = 3 \wedge voter(i).keyed = true \wedge$
$voter(i).finish = false \wedge$
$j = \xi^{-1}(\sigma_{Ad}^{-1}(\chi^{-1}(\sigma'_{Ad}(voter(i).cand), voter(i).b)), voter(i).r)$
Effect:
$voter(i).finish := true$

Figure 2: Transition rules of FOO

and for any other action a than the above, $f_V(a) = a$.

In the rest of this section, suppose i and i' are arbitrary voters. For any voter k, $\tilde{k}$ denotes i if $k = i'$, i' if $k = i$, and k otherwise.

Definition 4.2 The function $g_{i,i'}$ is defined as follows:
If $s.phase = 1$,
$g_{i,i'}(s).phase = s.phase$
$g_{i,i'}(s).line = s.line$
$g_{i,i'}(s).vbuff = s.vbuff$
$g_{i,i'}(s).kbuff = s.kbuff$
$g_{i,i'}(s).abuff = ex_{i,i'}(s.abuff)$
$g_{i,i'}(s).voter(k).sreq$
$= null$ if $s.voter(k).sreq = null$,
$\sigma'_{Ad}(s.voter(\tilde{k}).cand)$ otherwise.
$g_{i,i'}(s).voter(k).authed = s.voter(k).authed$ for any voter k.
$g_{i,i'}(s).voter(k).\psi = s.voter(\tilde{k}).\psi$ for any other subfield ψ of $voter(k)$ than $sreq$ and $authed$.

Otherwise, $g_{i,i'}$ simply exchanges i's fields for those of i'. Here, $ex_{i,i'}(x)$ is a multiset obtained by replacing each occurrence of $pair(k, \sigma_k(s.voter(k).cand))$ in x with $pair(k, \sigma_k(s.voter(\tilde{k}).cand))$.

The definition of the agent exchange above depends on the value of *phase* because the simulation policy must be switched during the execution of the protocol.

In phases 2 and 3, i's behavior can be mimicked by i' and vice versa since voters use the sender-anonymous communication channel for *cast* and *key* transitions.

Hence, $g_{i,i'}$ simply exchanges the states of i and i' in these phases.

In phase 1, however, i' cannot mimic i since the sender-anonymous channel is not used, and so i's behavior must be mimicked by i himself. Then, we cannot simply exchange subfields *authed* and *sreq* between i and i' since these concern the progress in phase 1, while subfields *cand*, b, r must be exchanged to maintain consistency with phases 2 and 3. And since these three subfields are exchanged, messages accumulated in *abuff* (and so the stored value of *sreq* after the *ack* action) must be exchanged between i and i'. Therefore, the above definition follows. Note that the observer cannot observe the values of *cand*, r and b of each voter since the messages in phase 1 are encrypted appropriately. Then we can prove the following theorem.

Theorem 4.3

1. *The functions $g_{i,i'}$ defined above for all distinct $i, i' \in I$ are agent exchange functions of S with respect to i, i' via f_V.*
2. *Any interpreted system that is ω-compatible with $traces(S)$ is role interchangeable.* ∎

4.4 Anonymity and Privacy of FOO

Recall that role interchangeability alone does not lead to anonymity or privacy in the sense of Section 2. We need some appropriate premise, which corresponds to a restriction of possible traces. What is an appropriate restriction for this problem?

A natural restriction is to consider only traces of *fair* executions with respect to $\Sigma_{I,A}$. Here, we say an execution $s_0, \delta_1, s_1, \delta_2, s_2, \ldots$ is fair with respect to $\Sigma_{I,A}$ if any action in $\Sigma_{I,A}$ continuously enabled will eventually be performed, that is, unless $i.a \in \Sigma_{I,A}$ and an integer n exist such that $i.a$ is possible in $s_{n'}$ and $\delta_{n'} \neq i.a$ for all $n' > n$. For instance, a finite execution is fair in this sense if and only if there is no possible action from $\Sigma_{I,A}$ in the final state. Let S be the automaton in Section 4.2. The set of all fair executions of S with respect to $\Sigma_{I,A}$ is denoted by $fexecs(S)$. We write $ftraces(S)$ instead of $trace(fexecs(S))$.

Another natural restriction is to assume trace $\hat{t}$ was obtained as the result of the actual execution of voting. For instance, if the total number of votes in the actual poll was 100, then we need not be concerned with the possibility that only one voter participated in the voting. With this assumption we can restrict ourselves to the traces t that satisfy $f_V(t) = f_V(\hat{t})$. We define the restriction $T \uparrow_{\hat{t}}^{f_V}$ of trace set T with trace $\hat{t} \in T$ and view function f_V as the set $\{t \in T \mid f_V(t) = f_V(\hat{t})\}$. It can be shown that this restriction preserves role interchangeability.

Then, we obtain the following verification result.

Theorem 4.4 *(Anonymity and Privacy of FOO)* *Suppose an interpreted system $\mathcal{I}$ is ω-compatible with $ftraces(S) \uparrow_{\hat{t}}^{f_V}$.*

Anonymity: Let I_A be the set of all voters who obtain the voting right in $\hat{t}$. Then, $vote(j)$ performed by i is anonymous up to I_A with respect to o in $\mathcal{I}$.

Privacy: Let $A_I = \{vote(j) \mid j$ is a candidate who wins a vote in $\hat{t}\}$. Then, any voter i who performs $vote(j)$ is private up to A_I with respect to o in $\mathcal{I}$. ∎

Note that the above theorem does not claim any anonymity for voters who did not obtain a voting right. In fact, a voter who did not obtain a voting right cannot interchange his role with a voter who obtained it since the observer knows who obtained it and who did not.

5 Conclusion

We formalized role interchangeability in multiagent systems, and showed how known anonymity and privacy properties are derived from it. We also presented a simulation proof method for the property. Then we applied our method to a verification of the anonymity and privacy of the FOO electronic voting protocol. We are now planning to extend our method to deal with probabilistic systems.

Acknowledgements

We thank Dr. Kiyoshi Shirayanagi and Prof. Masami Hagiya for helpful discussions.

References

[1] M. Abadi and C. Fournet. Private authentication. *TCS*, 322(3):427–476, 2004.

[2] D. L. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *CACM*, 24(2):84–90, 1981.

[3] A. Fujioka, T. Okamoto, and K. Ohta. A practical secret voting scheme for large scale elections. In *Advances in Cryptography–AUSCRYPT '92*, pages 244–251, 1992. LNCS 718.

[4] J. Y. Halpern and K. R. O'Neill. Anonymity and information hiding in multiagent systems. *JCS*, 13(3):483–514, 2005.

[5] D. Hughes and V. Shmatikov. Information hiding, anonymity and privacy: A modular approach. *JCS*, 12(1):3–36, 2004.

[6] Y. Kawabe, K. Mano, H. Sakurada, and Y. Tsukada. Simulation techniques to verify anonymity of security proptocols. In *Computer Security Symposium 2005*, volume I, pages 43–48, 2005. In Japanese.

[7] S. Kremer and M. Ryan. Analysis of an electronic voting protocol in the applied pi calculus. In *European Symposium on Programming–ESOP 2005*, pages 186–200, 2005. LNCS 3444.

[8] N. A. Lynch. *Distributed Algorithms*. Morgan Kaufmann Publishers, 1996.

[9] S. Schneider and A. Sidiropoulos. CSP and anonymity. In *Fourth European Symposium on Research in Computer Security (ESORICS 96)*, pages 198–218, 1996. LNCS 1146.